

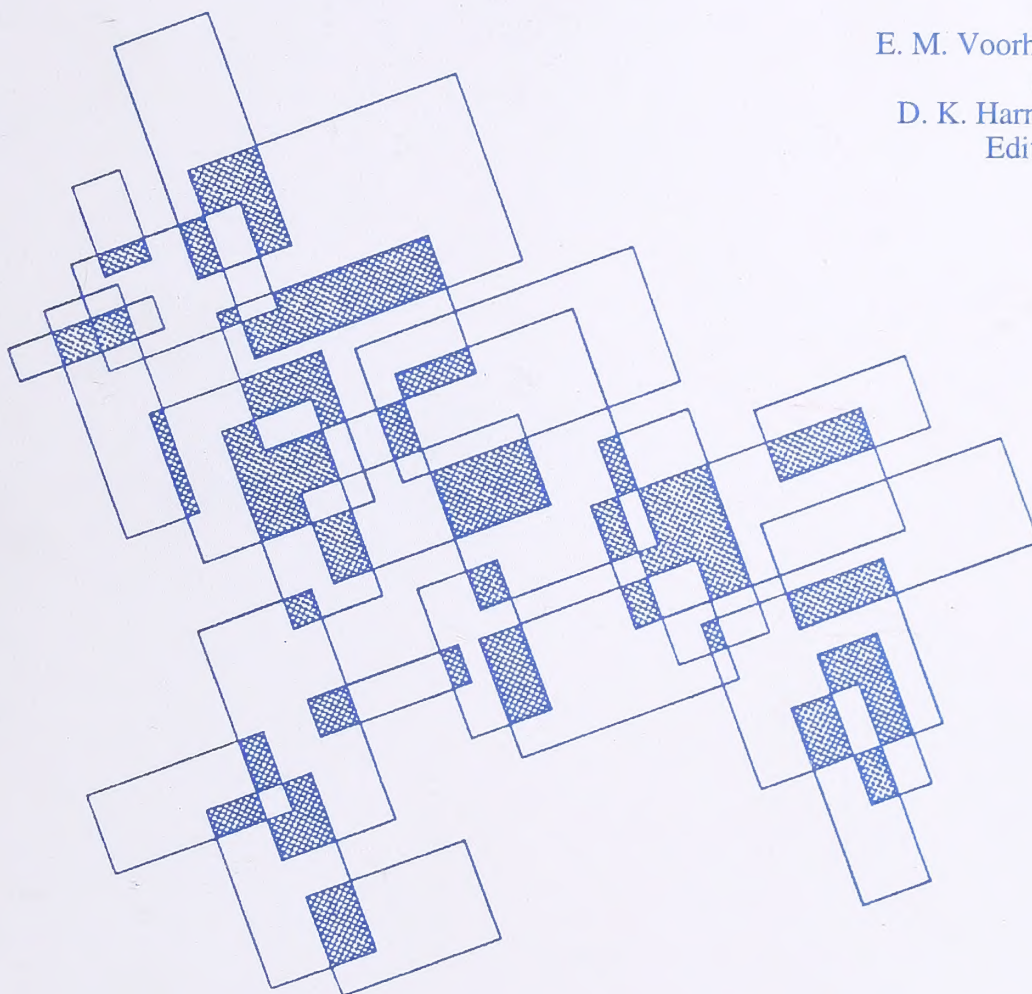


NIST Special Publication 500-246

Information Technology:

The Eighth Text REtrieval Conference (TREC-8)

E. M. Voorhees
and
D. K. Harman
Editors



NIST

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

QC
100
.U57
NO. 500-246
2000

The National Institute of Standards and Technology was established in 1988 by Congress to “assist industry in the development of technology . . . needed to improve product quality, to modernize manufacturing processes, to ensure product reliability . . . and to facilitate rapid commercialization . . . of products based on new scientific discoveries.”

NIST, originally founded as the National Bureau of Standards in 1901, works to strengthen U.S. industry's competitiveness; advance science and engineering; and improve public health, safety, and the environment. One of the agency's basic functions is to develop, maintain, and retain custody of the national standards of measurement, and provide the means and methods for comparing standards used in science, engineering, manufacturing, commerce, industry, and education with the standards adopted or recognized by the Federal Government.

As an agency of the U.S. Commerce Department's Technology Administration, NIST conducts basic and applied research in the physical sciences and engineering, and develops measurement techniques, test methods, standards, and related services. The Institute does generic and precompetitive work on new and advanced technologies. NIST's research facilities are located at Gaithersburg, MD 20899, and at Boulder, CO 80303. Major technical operating units and their principal activities are listed below. For more information contact the Publications and Program Inquiries Desk, 301-975-3058.

Office of the Director

- National Quality Program
- International and Academic Affairs

Technology Services

- Standards Services
- Technology Partnerships
- Measurement Services
- Information Services

Advanced Technology Program

- Economic Assessment
- Information Technology and Applications
- Chemistry and Life Sciences
- Materials and Manufacturing Technology
- Electronics and Photonics Technology

Manufacturing Extension Partnership Program

- Regional Programs
- National Programs
- Program Development

Electronics and Electrical Engineering Laboratory

- Microelectronics
- Law Enforcement Standards
- Electricity
- Semiconductor Electronics
- Radio-Frequency Technology¹
- Electromagnetic Technology¹
- Optoelectronics¹

Materials Science and Engineering Laboratory

- Intelligent Processing of Materials
- Ceramics
- Materials Reliability¹
- Polymers
- Metallurgy
- NIST Center for Neutron Research

Chemical Science and Technology Laboratory

- Biotechnology
- Physical and Chemical Properties²
- Analytical Chemistry
- Process Measurements
- Surface and Microanalysis Science

Physics Laboratory

- Electron and Optical Physics
- Atomic Physics
- Optical Technology
- Ionizing Radiation
- Time and Frequency¹
- Quantum Physics¹

Manufacturing Engineering Laboratory

- Precision Engineering
- Automated Production Technology
- Intelligent Systems
- Fabrication Technology
- Manufacturing Systems Integration

Building and Fire Research Laboratory

- Applied Economics
- Structures
- Building Materials
- Building Environment
- Fire Safety Engineering
- Fire Science

Information Technology Laboratory

- Mathematical and Computational Sciences²
- Advanced Network Technologies
- Computer Security
- Information Access and User Interfaces
- High Performance Systems and Services
- Distributed Computing and Information Services
- Software Diagnostics and Conformance Testing
- Statistical Engineering

¹At Boulder, CO 80303.

²Some elements at Boulder, CO.

NIST Special Publication 500-246

**Information Technology:
The Eighth Text REtrieval
Conference (TREC-8)**

E.M. Voorhees and
D.K. Harman
Editors

*Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20890-8980*

November 2000



U.S. Department of Commerce
Norman Y. Mineta, Secretary

Technology Administration
Dr. Cheryl L. Shavers, Under Secretary of Commerce for Technology

National Institute of Standards and Technology
Raymond G. Kammer, Director

Reports on Information Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) stimulates U.S. economic growth and industrial competitiveness through technical leadership and collaborative research in critical infrastructure technology, including tests, test methods, reference data, and forward-looking standards, to advance the development and productive use of information technology. To overcome barriers to usability, scalability, interoperability, and security in information systems and networks, ITL programs focus on a broad range of networking, security, and advanced information technologies, as well as the mathematical, statistical, and computational sciences. This Special Publication 500-series reports on ITL's research in tests and test methods for information technology, and its collaborative activities with industry, government, and academic organizations.

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

National Institute of Standards and Technology
Special Publication 500-246
Natl. Inst. Stand. Technol.
Spec. Publ. 500-246
1146 pages (November 2000)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 2000

For sale by the Superintendent of Documents
U.S. Government Printing Office
Washington, DC 20402-9325

Foreword

This report constitutes the proceedings of the eighth Text REtrieval Conference (TREC-8) held in Gaithersburg, Maryland, November 16–19, 1999. The conference was co-sponsored by the National Institute of Standards and Technology (NIST) and the Defense Advanced Research Projects Agency (DARPA), and was attended by 170 people. Sixty-six groups including participants from 16 different countries were represented. The conference was the eighth in an on-going series of workshops to evaluate new technologies in text retrieval.

The workshop included plenary sessions, discussion groups, a poster session, and demonstrations. Because the participants in the workshop drew on their personal experiences, they sometimes cited specific vendors and commercial products. The inclusion or omission of a particular company or product implies neither endorsement nor criticism by NIST. Any opinions, findings, and conclusions or recommendations expressed in the individual papers are the authors' own and do not necessarily reflect those of the sponsors.

The sponsorship of the Defense Advanced Research Projects Agency is gratefully acknowledged, as is the tremendous work of the program committee and the track coordinators.

Ellen Voorhees,
Donna Harman
September 19, 2000

TREC-8 Program Committee

Ellen Voorhees, NIST, chair
James Allan, University of Massachusetts at Amherst
Nick Belkin, Rutgers University
Chris Buckley, Sabir Research, Inc.
Jamie Callan, Carnegie Mellon University
Susan Dumais, Microsoft
Donna Harman, NIST
David Hawking, CSIRO
Bill Hersh, Oregon Health Sciences Institute
Darryl Howard, U.S. Department of Defense
David Hull, Xerox Research Centre Europe
John Prange, U.S. Department of Defense
Steve Robertson, Microsoft
Peter Schäuble, Swiss Federal Institute of Technology (ETH)
Amit Singhal, AT&T Labs-Research
Karen Sparck Jones, University of Cambridge, UK
Tomek Strzalkowski, GE Corporate Research and Development
Ross Wilkinson, CSIRO

TABLE OF CONTENTS

Alphabetical Index of TREC-8 Papers by Organization.....	xiii
Index of TREC-8 Papers by Task/Track	xix
Abstract	xxix

PAPERS

1. Overview of the Eighth Text REtrieval Conference (TREC-8)	1
E. M. Voorhees, D. K. Harman (National Institute of Standards and Technology)	
2. Cross-Language Information Retrieval (CLIR) Track Overview.....	25
M. Braschler, P. Schäuble (Eurospider Information Tech. AG)	
C. Peters (Istituto Elaborazione Informazione (CNR))	
3. The TREC-8 Filtering Track Final Report.....	35
D. A. Hull (Xerox Research Centre Europe)	
S. Robertson (Microsoft Research, UK)	
4. TREC-8 Interactive Track Report	57
W. Hersh (Oregon Health Sciences University)	
P. Over (National Institute of Standards and Technology)	
5. The TREC-8 Query Track	65
C. Buckley, J. Walz (Sabir Research, Inc.)	
6. The TREC-8 Question Answering Track Report.....	77
E. M. Voorhees (National Institute of Standards and Technology)	
7. The TREC-8 Question Answering Track Evaluation.....	83
E. M. Voorhees, D. M. Tice (National Institute of Standards and Technology)	
8. The TREC Spoken Document Retrieval Track: A Success Story	107
J. S. Garofolo, C. G. P. Auzanne, E. M. Voorhees	
(National Institute of Standards and Technology)	
9. Overview of the TREC-8 Web Track.....	131
D. Hawking (CSIRO Mathematical and Information Sciences)	
E. Voorhees (National Institute of Standards and Technology)	
N. Craswell, P. Bailey (Department of Computer Science, ANU)	
10. Okapi/Keenbow at TREC-8	151
S. E. Robertson (Microsoft Research Ltd., UK and City University, London, UK)	
S. Walker (Microsoft Research Ltd., UK)	

11. The Weaver System for Document Retrieval	163
A. Berger, J. Lafferty (Carnegie Mellon University)	
12. LASSO: A Tool for Surfing the Answer Net	175
D. Moldovan, S. Harabagiu, M. Paşca, R. Mihalcea, R. Goodrum, R. Gîrju, V. Rus (Southern Methodist University)	
13. Information Extraction Supported Question Answering	185
R. Srihari, W. Li (Cymfony Inc.)	
14. Spoken Document Retrieval For TREC-8 At Cambridge University	197
S. E. Johnson, P. C. Woodland (Cambridge University Engineering Department) P. Jourlin, K. Spärck Jones (Cambridge University Computer Laboratory)	
15. Oracle at Trec8: A Lexical Approach	207
K. Mahesh, J. Kud, P. Dixon (Oracle Corporation)	
16. TREC-8 Ad-Hoc, Query and Filtering Track Experiments using PIRCS.....	217
K.L. Kwok, L. Grunfeld, M. Chan (Queens College, CUNY)	
17. Report on the TREC-8 Experiment: Searching on the Web and in Distributed Collections	229
J. Savoy, J. Picard (Université de Neuchâtel, Switzerland)	
18. PLIERS at TREC8	241
A. MacFarlane, S. E. Robertson (City University, London, and Microsoft Research Ltd., UK) J. A. McCann (City University, London)	
19. Optimization in CLARIT TREC-8 Adaptive Filtering	253
C. Zhai, P. Jansen, N. Roma, E. Stoica, D.A. Evans (CLARITECH Corporation)	
20. Filters, Webs and Answers: The University of Iowa TREC-8 Results.....	259
D. Eichmann, P. Srinivasan (University of Iowa)	
21. DSO at TREC-8: A Hybrid Algorithm for the Routing Task.....	267
H. T. Ng, H. T. Ang, W. M. Soon (DSO National Laboratories)	
22. Fujitsu Laboratories TREC8 Report – Ad hoc, Small Web, and Large Web Track –	275
I. Namba, N. Igata (Fujitsu Laboratories Ltd.)	
23. Twenty-One at TREC-8: using Language Technology for Information Retrieval.....	285
W. Kraaij, R. Pohlmann (TNO-TPD) D. Hiemstra (University of Twente, CTIT)	

24. English-German Cross-Language Retrieval for the GIRT Collection – Exploiting a Multilingual Thesaurus.....	301
F.C. Gey, H. Jiang (University of California, Berkeley)	
25. ACSys TREC-8 Experiments	307
D. Hawking (CSIRO Mathematics and Information Sciences)	
P. Bailey, N. Craswell (Department of Computer Science, ANU)	
26. AT&T at TREC-8.....	317
A. Singhal, S. Abney, M. Bacchiani, M. Collins, D. Hindle, F. Pereira (AT&T Labs)	
27. CMU Spoken Document Retrieval in Trec-8: Analysis of the role of Term Frequency TF	331
M. Siegler, R. Jin, A. Hauptmann (Carnegie Mellon University)	
28. CLARIT TREC-8 Manual Ad-Hoc Experiments	335
D. A. Evans, J. Bennett, X. Tong, A. Huettner,	
C. Zhai, E. Stoica (CLARITECH Corporation)	
29. CLARIT TREC-8 CLIR Experiments	341
Y. Qu, H. Jin, A. N. Eilerman, E. Stoica, D. A. Evans (CLARITECH Corporation)	
30. CLARIT TREC-8 Experiments in Searching Web Data.....	345
J. Bennett, X. Tong, D. A. Evans (CLARITECH Corporation)	
31. Question-Answering Using Semantic Relation Triples	349
K. C. Litkowski (CL Research)	
32. A Connectivity Analysis Approach to Increasing Precision in Retrieval From Hyperlinked Documents.....	357
C. Gurrin, A. F. Smeaton (Dublin City University)	
33. The Eurospider Retrieval System and the TREC-8 Cross-Language Track	367
M. Braschler, P. Schäuble (Eurospider Information Technology AG)	
M-Y Kan, J. L. Klavans (Columbia University)	
34. TREC-8 Automatic Ad-Hoc Experiments at Fondazione Ugo Bordoni	377
C. Carpineto, G. Romano (Fondazione Ugo Bordoni)	
35. Natural Language Information Retrieval: TREC-8 Report	381
T. Strzalkowski (GE Research & Development)	
J. Perez-Carballo (Rutgers University)	
J. Karlgren (Swedish Institute of Computer Science)	
A. Hulth (Stockholm University)	
P. Tapanainen, T. Lahtinen (Conexor OY, Helsinki)	

36. Ad hoc, Cross-language and Spoken Document Information Retrieval at IBM.....	391
M. Franz, J. S. McCarley, R. T. Ward (IBM T.J. Watson Research Center)	
37. The Use of Predictive Annotation for Question Answering in TREC8.....	399
J. Prager, D. Radev, E. Brown, A. Coden (IBM T.J. Watson Research Center)	
V. Sann (Columbia University)	
38. IIT at TREC-8: Improving Baseline Precision.....	411
M. C. McCabe (Advanced Analytic Tools)	
D. O. Holmes (NCR Corporation)	
K. L. Alford (U.S. Army)	
A. Chowdhury (IIT Research Institute)	
D. A. Grossman, O. Frieder (Illinois Institute of Technology)	
39. Automatic Query Feedback using Related Words.....	421
S. M. Rüger (Imperial College of Science, Technology and Medicine)	
40. Two-Step Feature Selection and Neural Network Classification for the TREC-8 Routing.....	425
M. Stricker (Informatique-CDC and ESPCI)	
F. Vichot, F. Wolinski (Informatique-CDC)	
G. Dreyfus (ESPCI)	
41. Mercure at trec8: Adhoc, Web, CLIR and Filtering tasks.....	431
M. Boughanem, C. Julien, J. Mothe, C. Soule-Dupuy (IRIT/SIG)	
42. The JHU/APL HAIRCUT System at TREC-8	445
J. Mayfield, P. McNamee, C. Piatko (The Johns Hopkins University)	
43. Novel Query Expansion Technique using Apriori Algorithm	453
A. Rungsawang, A. Tangpong, P. Laohawee, T. Khampachua (Kasetsart University)	
44. Experiments on the TREC-8 Filtering Track.....	457
K. Hoashi, K. Matsumoto, N. Inoue, K. Hashimoto (KDD R&D Laboratories, Inc.)	
45. QALC – the Question-Answering program of the Language and Cognition group at LIMSI-CNRS.....	465
O. Ferret, B. Grau, G. Illouz, C. Jacquemin, N. Masson (LIMSI-CNRS)	
46. The LIMSI SDR System for TREC-8.....	475
J-L Gauvain, Y. de Kercadio, L. Lamel, G. Adda (LIMSI-CNRS)	
47. A Maximum Likelihood Ratio Information Retrieval Model.....	483
K. Ng (Massachusetts Institute of Technology)	

48. High Selectivity and Accuracy with READWARE's Automated System of Knowledge Organization	493
T. Adi, O. K. Ewell, P. Adi (Management Information Technologies, Inc.)	
49. A Sys Called Qanda	499
E. Breck, J. Burger, L. Ferro, D. House, M. Light, I. Mani (The MITRE Corporation)	
50. Description of Preliminary Results to TREC-8 QA Task	507
C-J Lin, H-H Chen (National Taiwan University)	
51. CRL's TREC-8 Systems Cross-Lingual IR, and Q&A.....	513
B. Ogden, J. Cowie, E. Ludovik, H. Molina-Salgado, S. Nirenburg, N. Sharples, S. Sheremtyeva (New Mexico State University)	
52. NTT DATA: Overview of system approach at TREC-8 ad-hoc and question answering....	523
T. Takaki(NTT Data Corporation)	
53. Do Batch and User Evaluations Give the Same Results? An Analysis from the TREC-8 Interactive Track	531
W. Hersh, A. Turpin, S. Price, D. Kraemer, B. Chan, L. Sacherek, D. Olson (Oregon Health Sciences University)	
54. Structuring and expanding queries in the probabilistic model.....	541
O. Yasushi, M. Hiroko, N. Masumi, H. Sakiko (RICOH Co., Ltd.)	
55. The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8.....	549
M. Fuller, M. Kaszkiel, S. Kimberley, J. Zobel (RMIT) C. Ng (RMIT and Sharp Laboratories of Europe Ltd.) R. Wilkinson (CSIRO) M. Wu (RMIT and CSIRO)	
56. Relevance Feedback <i>versus</i> Local Context Analysis as Term Suggestion Devices: Rutgers' TREC-8 Interactive Track Experience.....	565
N. J. Belkin, J. Head, J. Jeng, D. Kelly, S. Lin, L. Lobash, S. Y. Park (Rutgers University) C. Cool (Queens College, CUNY) P. Savage-Knepshield, C. Sikora (Lucent Technologies)	
57. An Early DiscoWeb Prototype at TREC8.....	575
B.D. Davison, A. Gerasoulis, K. Kleisouris, Y. Lu, H-j Seo, J. Tian, S. Wang, W. Wang, B. Wu (Rutgers University)	
58. SMART in TREC 8.....	577
C. Buckley, J. Walz (SabIR Research, Inc.)	

59. SCAI TREC-8 Experiments	583
D-H Shin, Y-H Kim, S. Kim, J-H Eom, H-J Shin, B-T Zhang (Seoul National University)	
60. TREC-8 Experiments at SUNY at Buffalo	591
B. Han, R. Nagarajan, R. Srihari, M. Srikanth (State University of New York at Buffalo)	
61. CINDOR Conceptual Interlingua Document Retrieval: TREC-8 Evaluation	597
M. Ruiz, A. Diekema, P. Sheridan (MNIS-TextWise Labs)	
62. CLIR using a Probabilistic Translation Model based on Web Documents	607
J-Y Nie (Université de Montréal)	
63. Berkeley's TREC 8 Interactive Track Entry: Cheshire II and Zprise	613
R. R. Larson (University of California, Berkeley)	
64. TREC-8 Experiments at Maryland: CLIR, QA and Routing.....	623
D. W. Oard, J. Wang (University of Maryland, College Park, MD)	
D. Lin (University of Manitoba)	
I. Soboroff (University of Maryland, Baltimore County, MD)	
65. INQUERY and TREC-8	637
J. Allan, J. Callan, F-F Feng, D. Malin (University of Massachusetts)	
66. IRIS at TREC-8.....	645
K. Yang, K. Maglaughlin (University of North Carolina)	
67. Moving More Quickly toward Full Term Relations in Information Space	657
G. B. Newby (University of North Carolina at Chapel Hill)	
68. Retrieval Performance and Visual Dispersion of Query Sets	669
M. Rorvig (The University of North Texas)	
69. Ask Me Tomorrow: The NRC and University of Ottawa Question Answering System	675
J. Martin (National Research Council), C. Lankester (University of Ottawa)	
70. Using Coreference in Question Answering	685
T. S. Morton (University of Pennsylvania)	
71. Interactive Okapi at Sheffield – TREC-8.....	689
M. Beaulieu, H. Fowkes, N. Alemayehu, M. Sanderson (University of Sheffield)	
72. The THISL SDR System At TREC-8	6999
D. Abberley, S. Renals (University of Sheffield)	
D. Ellis (ICSI, USA)	
T. Robinson (University of Cambridge, UK and SoftSound, UK)	

73. University of Sheffield TREC-8 Q & A System.....	707
K. Humphreys, R. Gaizauskas, M. Hepple, M. Sanderson (University of Sheffield)	
74. University of Surrey Participation in TREC8: Weirdness Indexing for Logical Document Extrapolation and Retrieval (WILDER)	717
K. Ahmad, L. Gillam, L. Tostevin (University of Surrey)	
75. The Mirror DBMS at TREC-8.....	725
A.P. de Vries, D. Hiemstra (University of Twente)	
76. Fast Automatic Passage Ranking (MultiText Experiments for TREC-8)	735
G. V. Cormack, C. L. A. Clarke, D. I. E. Kisman (University of Waterloo)	
C. R. Palmer (Carnegie Mellon University)	
77. Xerox TREC-8 Question Answering Track Report.....	743
D. A. Hull (Xerox Research Centre Europe)	

APPENDICES

A. TREC-8 Results	A-1
Track/Task Runs Lists.....	A-2
Evaluation Techniques and Measures.....	A-17
Ad hoc task results	A-24
Cross-language track results.....	A-153
Filtering track results.....	A-198
Interactive track results.....	A-213
Query track results.....	A-234
QA track results	A-237
Spoken document retrieval track results	A-283
Small web track results.....	A-347
Large web track results	A-391
B. Summary Performance Comparisons TREC-2 Through TREC-8.....	B-1
K. Sparck Jones (University of Cambridge)	

ALPHABETICAL INDEX OF TREC-8 PAPERS BY ORGANIZATION

ACSys	
ACSys TREC-8 Experiments.....	307
Advanced Analytic Tools	
IIT at TREC-8: Improving Baseline Precision.....	411
AT&T	
AT&T at TREC-8.....	317
Australian National University	
ACSys TREC-8 Experiments.....	307
Overview of the TREC-8 Web Track	131
Cambridge University	
Spoken Document Retrieval For TREC-8 At Cambridge University.....	197
The THISL SDR System At TREC-8	699
Carnegie Mellon University	
CMU Spoken Document Retrieval in Trec-8: Analysis of the role of Term Frequency TF.....	331
Fast Automatic Passage Ranking (MultiText Experiments for TREC-8).....	735
The Weaver System for Document Retrieval	163
City University, London, UK	
Okapi/Keenbow at TREC-8	151
PLIERS at TREC8	241
CLARITECH Corporation	
Optimization in CLARIT TREC-8 Adaptive Filtering	253
CLARIT TREC-8 Manual Ad-Hoc Experiments.....	335
CLARIT TREC-8 CLIR Experiments.....	341
CLARIT TREC-8 Experiments in Searching Web Data	345
CL Research	
Question-Answering Using Semantic Relation Triples	349
Columbia University	
The Eurospider Retrieval System and the TREC-8 Cross-Language Track.....	367
The Use of Predictive Annotation for Question Answering in TREC8.....	399
Conexor OY, Helsinki	
Natural Language Information Retrieval: TREC-8 Report.....	381

CSIRO Mathematical and Information Sciences	
ACSys TREC-8 Experiments.....	307
Overview of the TREC-8 Web Track	131
The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8.....	549
Cymfony Inc.	
Information Extraction Supported Question Answering.....	185
DSO National Laboratories	
DSO at TREC-8: A Hybrid Algorithm for the Routing Task	267
Dublin City University	
A Connectivity Analysis Approach to Increasing Precision in Retrieval From Hyperlinked Documents	357
ESPCI	
Two-Step Feature Selection and Neural Network Classification for the TREC-8 Routing.....	425
Eurospider Information Technology AG	
Cross-Language Information Retrieval (CLIR) Track Overview	25
The Eurospider Retrieval System and the TREC-8 Cross-Language Track.....	367
Fondazione Ugo Bordon	
TREC-8 Automatic Ad-Hoc Experiments at Fondazione Ugo Bordon.....	377
Fujitsu Laboratories Ltd.	
Fujitsu Laboratories TREC8 Report – Ad hoc, Small Web, and Large Web Track –.....	275
GE Research & Development	
Natural Language Information Retrieval: TREC-8 Report.....	381
IBM T.J. Watson Research Center	
Ad hoc, Cross-language and Spoken Document Information Retrieval at IBM.....	391
The Use of Predictive Annotation for Question Answering in TREC8	399
ICSI, USA	
The THISL SDR System At TREC-8	699
Illinois Institute of Technology	
IIT at TREC-8: Improving Baseline Precision.....	411
IIT Research Institute	
IIT at TREC-8: Improving Baseline Precision.....	411
Imperial College of Science, Technology and Medicine	
Automatic Query Feedback using Related Words	421

Informatique-CDC	
Two-Step Feature Selection and Neural Network Classification for the TREC-8 Routing.....	425
IRIT/SIG	
Mercure at trec8: Adhoc, Web, CLIR and Filtering tasks	431
Istituto Elaborazione Informazione (CNR)	
Cross-Language Information Retrieval (CLIR) Track Overview	25
The Johns Hopkins University	
The JHU/APL HAIRCUT System at TREC-8.....	445
Kasetsart University	
Novel Query Expansion Technique using Apriori Algorithm	453
KDD R&D Laboratories, Inc.	
Experiments on the TREC-8 Filtering Track	457
LIMSI-CNRS	
QALC – the Question-Answering program of the Language and Cognition group at LIMSI-CNRS	465
The LIMSI SDR System for TREC-8	475
Lucent Technologies	
Relevance Feedback <i>versus</i> Local Context Analysis as Term Suggestion Devices: Rutgers' TREC-8 Interactive Track Experience	565
Management Information Technologies, Inc.	
High Selectivity and Accuracy with READWARE's Automated System of Knowledge Organization.....	493
Massachusetts Institute of Technology	
A Maximum Likelihood Ratio Information Retrieval Model	483
Microsoft Research, UK	
Okapi/Keenbow at TREC-8	151
PLIERS at TREC8	241
The TREC-8 Filtering Track Final Report.....	35
The MITRE Corporation	
A Sys Called Qanda	499

National Institute of Standards and Technology (NIST)	
Overview of the Eighth Text REtrieval Conference (TREC-8)	1
Overview of the TREC-8 Web Track	131
The TREC-8 Question Answering Track Evaluation	83
The TREC-8 Question Answering Track Report	77
The TREC Spoken Document Retrieval Track: A Success Story	107
TREC-8 Interactive Track Report	57
National Research Council	
Ask Me Tomorrow: The NRC and University of Ottawa Question Answering System	675
National Taiwan University	
Description of Preliminary Results to TREC-8 QA Task	507
NCR Corporation	
IIT at TREC-8: Improving Baseline Precision	411
New Mexico State University	
CRL's TREC-8 Systems Cross-Lingual IR, and Q&A	513
NTT Data Corporation	
NTT DATA: Overview of system approach at TREC-8 ad-hoc and question answering	523
Oracle Corporation	
Oracle at Trec8: A Lexical Approach	207
Oregon Health Sciences University	
Do Batch and User Evaluations Give the Same Results? An Analysis from the TREC-8 Interactive Track	531
TREC-8 Interactive Track Report	57
Queens College, CUNY	
Relevance Feedback <i>versus</i> Local Context Analysis as Term Suggestion Devices: Rutgers' TREC-8 Interactive Track Experience	565
TREC-8 Ad-Hoc, Query and Filtering Track Experiments using PIRCS	217
RICOH Co., Ltd.	
Structuring and expanding queries in the probabilistic model	541
RMIT	
The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8	549
Rutgers University	
An Early DiscoWeb Prototype at TREC8	575
Natural Language Information Retrieval: TREC-8 Report	381
Relevance Feedback <i>versus</i> Local Context Analysis as Term Suggestion Devices: Rutgers' TREC-8 Interactive Track Experience	565

SabIR Research, Inc.	
SMART in TREC 8	577
The TREC-8 Query Track	65
Seoul National University	
SCAI TREC-8 Experiments	583
Sharp Laboratories of Europe Ltd.	
The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8	549
SoftSound, UK	
The THISL SDR System At TREC-8	699
Southern Methodist University	
LASSO: A Tool for Surfing the Answer Net	175
State University of New York at Buffalo	
TREC-8 Experiments at SUNY at Buffalo	591
Stockholm University	
Natural Language Information Retrieval: TREC-8 Report	381
Swedish Institute of Computer Science	
Natural Language Information Retrieval: TREC-8 Report	381
MNIS-TextWise Labs	
CINDOR Conceptual Interlingua Document Retrieval: TREC-8 Evaluation	597
TNO-TPD	
Twenty-One at TREC-8: using Language Technology for Information Retrieval	285
Université de Montréal	
CLIR using a Probabilistic Translation Model based on Web Documents	607
Université de Neuchâtel, Switzerland	
Report on the TREC-8 Experiment: Searching on the Web and in Distributed Collections	229
University of California, Berkeley	
Berkeley's TREC 8 Interactive Track Entry: Cheshire II and Zprise	613
English-German Cross-Language Retrieval for the GIRT Collection – Exploiting a	
Multilingual Thesaurus	301
University of Iowa	
Filters, Webs and Answers: The University of Iowa TREC-8 Results	259
University of Manitoba	
TREC-8 Experiments at Maryland: CLIR, QA and Routing	623

University of Maryland, Baltimore County, MD	
TREC-8 Experiments at Maryland: CLIR, QA and Routing	623
University of Maryland, College Park, MD	
TREC-8 Experiments at Maryland: CLIR, QA and Routing	623
University of Massachusetts	
INQUERY and TREC-8.....	637
University of North Carolina	
IRIS at TREC-8.....	645
University of North Carolina at Chapel Hill	
Moving More Quickly toward Full Term Relations in Information Space.....	657
The University of North Texas	
Retrieval Performance and Visual Dispersion of Query Sets	669
University of Ottawa	
Ask Me Tomorrow: The NRC and University of Ottawa Question Answering System	675
University of Pennsylvania	
Using Coreference in Question Answering.....	685
University of Sheffield	
Interactive Okapi at Sheffield – TREC-8	689
The THISL SDR System At TREC-8	699
University of Sheffield TREC-8 Q & A System.....	707
University of Surrey	
University of Surrey Participation in TREC8: Weirdness Indexing for Logical Document Extrapolation and Retrieval (WILDER)	717
University of Twente	
The Mirror DBMS at TREC-8	725
Twenty-One at TREC-8: using Language Technology for Information Retrieval	285
University of Waterloo (MultiText)	
Fast Automatic Passage Ranking (MultiText Experiments for TREC-8).....	735
US Army	
IIT at TREC-8: Improving Baseline Precision.....	411
Xerox Research Centre Europe	
The TREC-8 Filtering Track Final Report	35
Xerox TREC-8 Question Answering Track Report	743

INDEX OF TREC-8 PAPERS BY TASK/TRACK

AD HOC

ACSys/CSIRO Mathematics and Information Sciences/Australian National University	
ACSys TREC-8 Experiments.....	307
AT&T	
AT&T at TREC-8.....	317
Carnegie Mellon University	
The Weaver System for Document Retrieval	163
City University, London, UK/Microsoft Research Ltd., UK	
PLIERS at TREC8	241
CLARITECH Corporation	
CLARIT TREC-8 Manual Ad-Hoc Experiments.....	335
Fondazione Ugo Bordonì	
TREC-8 Automatic Ad-Hoc Experiments at Fondazione Ugo Bordonì.....	377
Fujitsu Laboratories Ltd.	
Fujitsu Laboratories TREC8 Report – Ad hoc, Small Web, and Large Web Track –.....	275
GE Research & Development/Rutgers University/Swedish Institute of Computer Science/Stockholm University/ Conexor OY, Helsinki	
Natural Language Information Retrieval: TREC-8 Report.....	381
IBM T.J. Watson Research Center	
Ad hoc, Cross-language and Spoken Document Information Retrieval at IBM.....	391
IBM T.J. Watson Research Center/Columbia University	
The Use of Predictive Annotation for Question Answering in TREC8.....	399
Illinois Institute of Technology/Advanced Analytic Tools/NCR Corporation/U.S. Army/IIT Research Institute	
IIT at TREC-8: Improving Baseline Precision.....	411
Imperial College of Science, Technology and Medicine	
Automatic Query Feedback using Related Words	421
IRIT/SIG	
Mercure at trec8: Adhoc, Web, CLIR and Filtering tasks	431

The Johns Hopkins University	
The JHU/APL HAIRCUT System at TREC-8	445
Kasetsart University	
Novel Query Expansion Technique using Apriori Algorithm	453
KDD R&D Laboratories, Inc.	
Experiments on the TREC-8 Filtering Track	457
Management Information Technologies, Inc.	
High Selectivity and Accuracy with READWARE's Automated System of Knowledge Organization.....	493
Massachusetts Institute of Technology	
A Maximum Likelihood Ratio Information Retrieval Model	483
Microsoft Research Ltd., UK/City University, London, UK	
Okapi/Keenbow at TREC-8	151
MultiText/University of Waterloo/Carnegie Mellon University	
Fast Automatic Passage Ranking (MultiText Experiments for TREC-8).....	735
NTT Data Corporation	
NTT DATA: Overview of system approach at TREC-8 ad-hoc and question answering	523
Oracle Corporation	
Oracle at Trec8: A Lexical Approach	207
Queens College, CUNY	
TREC-8 Ad-Hoc, Query and Filtering Track Experiments using PIRCS.....	217
RICOH Co., Ltd.	
Structuring and expanding queries in the probabilistic model.....	541
RMIT/Sharp Laboratories of Europe Ltd./CSIRO	
The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8	549
Rutgers University	
An Early DiscoWeb Prototype at TREC8.....	575

SabIR Research, Inc.	
SMART in TREC 8.....	577
Seoul National University	
SCAI TREC-8 Experiments	583
State University of New York at Buffalo	
TREC-8 Experiments at SUNY at Buffalo	591
TNO-TPD/University of Twente, CTIT	
Twenty-One at TREC-8: using Language Technology for Information Retrieval	285
Université de Neuchâtel, Switzerland	
Report on the TREC-8 Experiment: Searching on the Web and in Distributed Collections	229
University of Maryland, College Park, MD/University of Manitoba/University of Maryland, Baltimore County, MD	
TREC-8 Experiments at Maryland: CLIR, QA and Routing	623
University of Massachusetts	
INQUERY and TREC-8.....	637
University of North Carolina	
IRIS at TREC-8	645
University of North Carolina at Chapel Hill	
Moving More Quickly toward Full Term Relations in Information Space.....	657
The University of North Texas	
Retrieval Performance and Visual Dispersion of Query Sets	669
University of Surrey	
University of Surrey Participation in TREC8: Weirdness Indexing for Logical Document Extrapolation and Retrieval (WILDER)	717
University of Twente	
The Mirror DBMS at TREC-8	725

CROSS-LANGUAGE

CLARITECH Corporation	
CLARIT TREC-8 CLIR Experiments.....	341
Eurospider Information Technology AG/Columbia University	
The Eurospider Retrieval System and the TREC-8 Cross-Language Track.....	367

Eurospider Information Technology AG/Istituto Elaborazione Informazione (CNR)	
Cross-Language Information Retrieval (CLIR) Track Overview	25
IBM T.J. Watson Research Center	
Ad hoc, Cross-language and Spoken Document Information Retrieval at IBM.....	391
IRIT/SIG	
Mercure at trec8: Adhoc, Web, CLIR and Filtering tasks	431
The Johns Hopkins University	
The JHU/APL HAIRCUT System at TREC-8.....	445
New Mexico State University	
CRL's TREC-8 Systems Cross-Lingual IR, and Q&A.....	513
MNIS-TextWise Labs	
CINDOR Conceptual Interlingua Document Retrieval: TREC-8 Evaluation	597
TNO-TPD/University of Twente, CTIT	
Twenty-One at TREC-8: using Language Technology for Information Retrieval	285
Université de Montréal	
CLIR using a Probabilistic Translation Model based on Web Documents.....	607
University of California, Berkeley	
English-German Cross-Language Retrieval for the GIRT Collection – Exploiting a Multilingual Thesaurus	301
University of Maryland, College Park, MD/University of Manitoba/University of Maryland, Baltimore County, MD	
TREC-8 Experiments at Maryland: CLIR, QA and Routing	623

FILTERING

City University, London, UK/Microsoft Research Ltd., UK	
PLIERS at TREC8	241
CLARITECH Corporation	
Optimization in CLARIT TREC-8 Adaptive Filtering	253
DSO National Laboratories	
DSO at TREC-8: A Hybrid Algorithm for the Routing Task	267
Informatique-CDC/ESPCI	
Two-Step Feature Selection and Neural Network Classification for the TREC-8 Routing.....	425

IRIT/SIG

Mercure at trec8: Adhoc, Web, CLIR and Filtering tasks	431
--	-----

KDD R&D Laboratories, Inc.

Experiments on the TREC-8 Filtering Track	457
---	-----

Microsoft Research Ltd., UK/City University, London, UK

Okapi/Keenbow at TREC-8	151
-------------------------------	-----

Queens College, CUNY

TREC-8 Ad-Hoc, Query and Filtering Track Experiments using PIRCS.....	217
---	-----

Seoul National University

SCAI TREC-8 Experiments	583
-------------------------------	-----

TNO-TPD/ University of Twente, CTIT

Twenty-One at TREC-8: using Language Technology for Information Retrieval	285
---	-----

University of Iowa

Filters, Webs and Answers: The University of Iowa TREC-8 Results	259
--	-----

University of Maryland, College Park, MD/University of Manitoba/University of Maryland, Baltimore County, MD

TREC-8 Experiments at Maryland: CLIR, QA and Routing	623
--	-----

University of Massachusetts

INQUERY and TREC-8.....	637
-------------------------	-----

Xerox Research Centre Europe/Microsoft Research, UK

The TREC-8 Filtering Track Final Report.....	35
--	----

INTERACTIVE

Oregon Health Sciences University

Do Batch and User Evaluations Give the Same Results? An Analysis from the TREC-8 Interactive Track	531
--	-----

Oregon Health Sciences University/National Institute of Standards and Technology

TREC-8 Interactive Track Report	57
---------------------------------------	----

RMIT/Sharp Laboratories of Europe Ltd./CSIRO

The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8	549
--	-----

Rutgers University/Queens College, CUNY/Lucent Technologies

Relevance Feedback <i>versus</i> Local Context Analysis as Term Suggestion Devices: Rutgers'	565
--	-----

University of California, Berkeley	
Berkeley's TREC 8 Interactive Track Entry: Cheshire II and Zprise	613
University of North Carolina	
IRIS at TREC-8.....	645
University of Sheffield	
Interactive Okapi at Sheffield – TREC-8	689

QUERY

ACSys/CSIRO Mathematics and Information Sciences/Australian National University	
ACSys TREC-8 Experiments.....	307
Queens College, CUNY	
TREC-8 Ad-Hoc, Query and Filtering Track Experiments using PIRCS.....	217
SabIR Research, Inc.	
SMART in TREC 8.....	577
The TREC-8 Query Track.....	65
University of Massachusetts	
INQUERY and TREC-8.....	637

QUESTION ANSWERING

AT&T	
AT&T at TREC-8.....	317
CL Research	
Question-Answering Using Semantic Relation Triples	349
Cymfony Inc.	
Information Extraction Supported Question Answering.....	185
GE Research & Development/Rutgers University/Swedish Institute of Computer Science/Stockholm University/ Conexor OY, Helsinki	
Natural Language Information Retrieval: TREC-8 Report.....	381
IBM T.J. Watson Research Center/Columbia University	
The Use of Predictive Annotation for Question Answering in TREC8.....	399
LIMSI-CNRS	
QALC – the Question-Answering program of the Language and Cognition group at LIMSI-CNRS	465

The MITRE Corporation	
A Sys Called Qanda	599
MultiText/University of Waterloo/Carnegie Mellon University	
Fast Automatic Passage Ranking (MultiText Experiments for TREC-8)	735
National Institute of Standards and Technology (NIST)	
The TREC-8 Question Answering Track Evaluation	83
The TREC-8 Question Answering Track Report	77
National Taiwan University	
Description of Preliminary Results to TREC-8 QA Task	507
New Mexico State University	
CRL's TREC-8 Systems Cross-Lingual IR, and Q&A	513
NTT Data Corporation	
NTT DATA: Overview of system approach at TREC-8 ad-hoc and question answering	523
RMIT/Sharp Laboratories of Europe Ltd./CSIRO	
The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8	549
Seoul National University	
SCAI TREC-8 Experiments	583
Southern Methodist University	
LASSO: A Tool for Surfing the Answer Net	175
University of Iowa	
Filters, Webs and Answers: The University of Iowa TREC-8 Results	259
University of Maryland, College Park, MD/University of Manitoba/University of Maryland, Baltimore County, MD	
TREC-8 Experiments at Maryland: CLIR, QA and Routing	623
University of Massachusetts	
INQUERY and TREC-8	637
University of Ottawa/National Research Council	
Ask Me Tomorrow: The NRC and University of Ottawa Question Answering System	675
University of Pennsylvania	
Using Coreference in Question Answering	685

University of Sheffield	
University of Sheffield TREC-8 Q & A System.....	707
Xerox Research Centre Europe	
Xerox TREC-8 Question Answering Track Report	743

SPOKEN DATA RECOGNITION

AT&T	
AT&T at TREC-8.....	317
Cambridge University	
Spoken Document Retrieval For TREC-8 At Cambridge University.....	197
Carnegie Mellon University	
CMU Spoken Document Retrieval in Trec-8: Analysis of the role of Term Frequency TF.....	331
IBM T.J. Watson Research Center	
Ad hoc, Cross-language and Spoken Document Information Retrieval at IBM.....	391
LIMSI-CNRS	
The LIMSI SDR System for TREC-8	475
National Institute of Standards and Technology	
The TREC Spoken Document Retrieval Track: A Success Story	107
RMIT/Sharp Laboratories of Europe Ltd./CSIRO	
The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8	549
State University of New York at Buffalo	
TREC-8 Experiments at SUNY at Buffalo	591
TNO-TPD/ University of Twente, CTIT	
Twenty-One at TREC-8: using Language Technology for Information Retrieval	285
University of Massachusetts	
INQUERY and TREC-8.....	637
University of Sheffield/ICSI, USA/University of Cambridge, UK/SoftSound, UK	
The THISL SDR System At TREC-8	699

SMALL WEB

ACSys/CSIRO Mathematical and Information Sciences/Australian National University	
ACSys TREC-8 Experiments.....	307
AT&T	
AT&T at TREC-8.....	317
CLARITECH Corporation	
CLARIT TREC-8 Experiments in Searching Web Data	345
CSIRO Mathematical and Information Sciences/National Institute of Standards and Technology/Australian National University	
Overview of the TREC-8 Web Track	131
Dublin City University	
A Connectivity Analysis Approach to Increasing Precision in Retrieval From Hyperlinked Documents	357
Fujitsu Laboratories Ltd.	
Fujitsu Laboratories TREC8 Report – Ad hoc, Small Web, and Large Web Track –.....	275
Illinois Institute of Technology/Advanced Analytic Tools/NCR Corporation/U.S. Army/IIT Research Institute	
IIT at TREC-8: Improving Baseline Precision.....	411
IRIT/SIG	
Mercure at trec8: Adhoc, Web, CLIR and Filtering tasks	431
Microsoft Research Ltd., UK/City University, London, UK	
Okapi/Keenbow at TREC-8	151
MultiText/University of Waterloo/Carnegie Mellon University	
Fast Automatic Passage Ranking (MultiText Experiments for TREC-8).....	735
RMIT/Sharp Laboratories of Europe Ltd./CSIRO	
The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8	549
Seoul National University	
SCAI TREC-8 Experiments	583
Université de Neuchâtel, Switzerland	
Report on the TREC-8 Experiment: Searching on the Web and in Distributed Collections	229
University of Iowa	
Filters, Webs and Answers: The University of Iowa TREC-8 Results	259

University of Massachusetts	
INQUERY and TREC-8.....	637

University of North Carolina at Chapel Hill	
Moving More Quickly toward Full Term Relations in Information Space.....	657

LARGE WEB

ACSys/CSIRO Mathematical and Information Sciences/Australian National University	
ACSys TREC-8 Experiments.....	307

AT&T	
AT&T at TREC-8.....	317

City University, London, UK/Microsoft Research Ltd., UK	
PLIERS at TREC8	241

CSIRO Mathematical and Information Sciences/National Institute of Standards and Technology/Australian National University	
Overview of the TREC-8 Web Track	131

Fujitsu Laboratories Ltd.	
Fujitsu Laboratories TREC8 Report – Ad hoc, Small Web, and Large Web Track –.....	275

Microsoft Research Ltd., UK/City University, London, UK	
Okapi/Keenbow at TREC-8	151

MultiText/University of Waterloo/Carnegie Mellon University	
Fast Automatic Passage Ranking (MultiText Experiments for TREC-8).....	735

Rutgers University	
An Early DiscoWeb Prototype at TREC8.....	575

University of Massachusetts	
INQUERY and TREC-8.....	637

University of North Carolina at Chapel Hill	
Moving More Quickly toward Full Term Relations in Information Space.....	657

Abstract

This report constitutes the proceedings of the eighth Text REtrieval Conference (TREC-8) held in Gaithersburg, Maryland, November 16–19, 1999. The conference was co-sponsored by the National Institute of Standards and Technology (NIST) and the Defense Advanced Research Projects Agency (DARPA), and was attended by 170 people. Sixty-six groups including participants from 16 different countries were represented.

The goal of the conference was to bring research groups together to discuss their work on a large test collection. The diversity of the participants meant that a wide variety of retrieval techniques were represented, including new models for retrieval and refined machine learning techniques. Results were scored using a common evaluation package, so groups were able to compare the effectiveness of different techniques, and to discuss how differences between systems affected performance. In addition to the main evaluation, seven additional evaluations, called “tracks,” allowed participants to focus on particular common subproblems. Two new tracks were introduced this year. The question answering track encouraged research into systems that return answers, rather than ranked lists of documents, in response to a question. The web track investigated retrieval performance over a collection of World Wide Web pages.

The conference included paper sessions and discussion groups. This proceedings includes papers from most of the participants (some groups did not submit papers), track reports that define the problem addressed by the track plus summarize the main track results, and tables of individual group results. The TREC-8 proceedings web site also contains system descriptions that detail the timing and storage requirements of the different runs.

Overview of the Eighth Text REtrieval Conference (TREC-8)

Ellen M. Voorhees, Donna Harman
National Institute of Standards and Technology
Gaithersburg, MD 20899

1 Introduction

The eighth Text REtrieval Conference (TREC-8) was held at the National Institute of Standards and Technology (NIST) on November 16–19, 1999. The conference was co-sponsored by NIST and the Information Technology Office of the Defense Advanced Research Projects Agency (DARPA).

TREC-8 is the latest in a series of workshops designed to foster research in text retrieval. For analyses of the results of previous workshops, see Tague-Sutcliffe and Blustein [11], Harman [4], and Sparck Jones [10]. In addition, the overview paper in each of the previous TREC proceedings summarizes the results of that TREC.

The TREC workshop series has the following goals:

- to encourage research in text retrieval based on large test collections;
- to increase communication among industry, academia, and government by creating an open forum for the exchange of research ideas;
- to speed the transfer of technology from research labs into commercial products by demonstrating substantial improvements in retrieval methodologies on real-world problems; and
- to increase the availability of appropriate evaluation techniques for use by industry and academia, including development of new evaluation techniques more applicable to current systems.

Table 1 lists the groups that participated in TREC-8. Sixty-six groups including participants from 16 different countries were represented. The diversity of the participating groups has ensured that TREC represents many different approaches to text retrieval. The emphasis on individual experiments evaluated within a common setting has proven to be a major strength of TREC.

This paper serves as an introduction to the research described in detail in the remainder of the volume. It concentrates on the main task, *ad hoc retrieval*, which is defined in the next section. Details regarding the test collections and evaluation methodology used in TREC follow in sections 3 and 4, while section 5 provides an overview of the ad hoc retrieval results. In addition to the main ad hoc task, TREC-8 contained seven “tracks,” tasks that focus research on particular subproblems of text retrieval. Taken together, the tracks represent the bulk of the experiments performed in TREC-8. However, each track has its own overview paper included in the proceedings, so this paper presents only a short summary of each track in section 6. The final section looks forward to future TREC conferences.

2 The Ad Hoc Retrieval Task

The ad hoc retrieval task investigates the performance of systems that search a static set of documents using new questions (called *topics* in TREC). This task is similar to how a researcher might use a library—the collection is known but the questions likely to be asked are not known. NIST provides the participants approximately 2 gigabytes worth of documents and a set of 50 natural language topic statements. The participants produce a set of *queries* from the topic statements and run those queries against the documents. The output from this run is the official test result for the ad hoc task. Participants return the best 1000 documents retrieved for each topic to NIST for evaluation.

Table 1: Organizations participating in TREC-8

ACSys	National Taiwan University
AT&T Labs Research	New Mexico State University
CL Research	Oracle
CLARITECH Corporation	Oregon Health Sciences University
Cambridge University	Oslo College
Carnegie Mellon University	Queens College, CUNY
Center for Information Research, Russia	RICOH Co., Ltd.
City University/Microsoft	RMIT
Cymfony Inc.	Rutgers University (3 groups)
DSO National Laboratories, Singapore	Sabir Research/Cornell University
Dartmouth College	Seoul National University
Dublin City University	Sharp Laboratories of Europe Ltd.
Eurospider Information Technology	Southern Methodist University
Fondazione Ugo Bordon	State University of New York at Buffalo
Fujitsu Laboratories, Ltd.	TextWise, Inc.
GE/Rutgers/SICS/UHelsinki/UPenn	The University of Sheffield, UK
IBM T. J. Watson Research Center (2 groups)	TwentyOne
IIT/AAT/NCR	USheffield/CambridgeU/SoftSound/ICSI Berkeley
IRIT/SIG	Universite de Montreal
Imperial College	Universite de Neuchatel
Informatique-CDC	University of California, Berkeley
Johns Hopkins University	University of Iowa
KDD R&D Laboratories	University of Maryland, College Park
Kasetsart University	University of Massachusetts
LIMSI-CNRS (2 groups)	University of North Carolina (2 groups)
MIT Laboratory for Computer Science	University of North Texas
MITRE	University of Ottawa
Management Information Technologies, Inc.	University of Surrey
Microsoft Research Ltd	University of Twente
MuliText Project	Xerox Research Centre Europe
NTT DATA Corporation	

Participants are free to use any method they desire to create the queries from the topic statements. TREC distinguishes among two major categories of query construction techniques, automatic methods and manual methods. An automatic method is a means of deriving a query from the topic statement with no manual intervention whatsoever; a manual method is anything else. The definition of manual query construction methods is very broad, ranging from simple tweaks to an automatically derived query, through manual construction of an initial query, to multiple query reformulations based on the document sets retrieved. Since these methods require radically different amounts of (human) effort, care must be taken when comparing manual results to ensure that the runs are truly comparable.

The right answers, called *relevance judgments*, for the ad hoc topics are not known at the time the participants produce their runs, though participants may use the documents, topics, and relevance judgments from previous TRECs to develop their systems. Participants are also free to use other sources of training data if they desire. Fifty new topics (401–450) were created for the TREC-8 ad hoc task. The set of documents used in the task was the documents contained on TREC Disks 4 and 5, excluding the *Congressional Record* subcollection. (See section 3.1 for details about this document set.) Disks 4 and 5 have now been used as the set of test documents for TRECs 6, 7, and 8 to produce a test collection with 150 topics.

Participants were allowed to submit up to five ad hoc runs to NIST. The runs could differ as the result of using different query construction techniques, or using different searching methods with the same queries.

Table 2: Document collection statistics. Words are strings of alphanumeric characters. No stop words were removed and no stemming was performed.

	Size (megabytes)	# Docs	Median # Words/Doc	Mean # Words/Doc
Disk 1				
<i>Wall Street Journal</i> , 1987–1989	267	98,732	245	434.0
<i>Associated Press</i> newswire, 1989	254	84,678	446	473.9
<i>Computer Selects</i> articles, Ziff-Davis	242	75,180	200	473.0
<i>Federal Register</i> , 1989	260	25,960	391	1315.9
abstracts of U.S. DOE publications	184	226,087	111	120.4
Disk 2				
<i>Wall Street Journal</i> , 1990–1992 (WSJ)	242	74,520	301	508.4
<i>Associated Press</i> newswire (1988) (AP)	237	79,919	438	468.7
<i>Computer Selects</i> articles, Ziff-Davis (ZIFF)	175	56,920	182	451.9
<i>Federal Register</i> (1988) (FR88)	209	19,860	396	1378.1
Disk 3				
<i>San Jose Mercury News</i> , 1991	287	90,257	379	453.0
<i>Associated Press</i> newswire, 1990	237	78,321	451	478.4
<i>Computer Selects</i> articles, Ziff-Davis	345	161,021	122	295.4
U.S. patents, 1993	243	6,711	4445	5391.0
Disk 4				
the <i>Financial Times</i> , 1991–1994 (FT)	564	210,158	316	412.7
<i>Federal Register</i> , 1994 (FR94)	395	55,630	588	644.7
<i>Congressional Record</i> , 1993 (CR)	235	27,922	288	1373.5
Disk 5				
Foreign Broadcast Information Service (FBIS)	470	130,471	322	543.6
the <i>LA Times</i>	475	131,896	351	526.5

When submitting a run, participants were required to state whether the queries were produced manually or automatically. If any run used an automatic method, participants were required to submit a run that used just the “title” and “description” fields of the topic statements (see section 3.2 for a description of the topics).

3 The Test Collections

Like most traditional retrieval collections, there are three distinct parts to the collections used in TREC: the documents, the topics, and the relevance judgments. This section describes each of these pieces for the ad hoc collection.

3.1 Documents

TREC documents are distributed on CD-ROM’s with approximately 1 GB of text on each, compressed to fit. For TREC-8, Disks 1–5 were all available as training material (see table 2) and Disks 4–5 were used for the ad hoc task. The *Congressional Record* subcollection on Disk 4 was excluded from the test document set.

Documents are tagged using SGML to allow easy parsing (see fig. 1). The documents in the different datasets have been tagged with identical major structures but they have different minor structures. The philosophy in the formatting at NIST is to leave the data as close to the original as possible. No attempt is made to correct spelling errors, sentence fragments, strange formatting around tables, or similar faults.

```

<DOC>
<DOCNO>FT911-3</DOCNO>
<PROFILE>AN-BEOA7AAIFT</PROFILE>
<DATE>910514
</DATE>
<HEADLINE>
FT 14 MAY 91 / International Company News: Contigas plans DM900m east German
project
</HEADLINE>
<BYLINE>
By DAVID GOODHART
</BYLINE>
<DATELINE>
BONN
</DATELINE>
<TEXT>
CONTIGAS, the German gas group 81 per cent owned by the utility Bayernwerk, said
yesterday that it intends to invest DM900m (Dollars 522m) in the next four years
to build a new gas distribution system in the east German state of Thuringia. ...
</TEXT>
</DOC>

```

Figure 1: A document extract from the *Financial Times*.

3.2 Topics

The format of the TREC topics has evolved over time as illustrated in table 3. The table shows the number of words included in the different parts of the topic statements for each TREC. The original ad hoc topics (51–150) were very detailed, containing multiple fields and lists of concepts related to the topic subject. The ad hoc topics used in TREC-3 (151–200) did not contain the concept lists and the remaining fields were generally shorter than in earlier topics. Nonetheless, participants in TREC-3 felt that the topics were still too long compared with what users normally submit to operational retrieval systems. The TREC-4 topics (201–250) were therefore made even shorter: a single field consisting of a one sentence description of the information need. However, the one-sentence topic eliminated from the topic the statement of the criteria used to judge a document as relevant—which was one of the motivating factors for providing topic statements rather than queries. The last four sets of ad hoc topics (251–450) have therefore all had the same format as in TREC-3, consisting of a title, description, and narrative. A sample TREC-8 topic is shown in figure 2.

The different parts in the most recent TREC topics allow participants to investigate the effect of different query lengths on retrieval performance. The “titles” in topics 301–450 have been specially designed to allow experiments with very short queries. The titles consist of up to three words that best describe the topic. The description field is a one sentence description of the topic area. As in TREC-7, the description field of TREC-8 topics contains all of the words in the title field, to remove the confounding effects of word choice on length experiments. The narrative gives a concise description of what makes a document relevant.

Ad hoc topics have been constructed by the same person who performed the relevance assessments for that topic (called the *assessor*) since TREC-3. Each assessor comes to NIST with ideas for topics based on his or her own interests, and searches the ad hoc collection (looking at approximately 100 documents per topic) to estimate the likely number of relevant documents per candidate topic. Because the same topic set was also to be used in the Web track this year, once the assessors searched the ad hoc collection with a candidate topic, they also searched the small Web collection using that topic. The NIST TREC team selected the final 50 topics from among the candidates based on the estimated number of relevant documents

Table 3: Topic length statistics by topic section. Lengths count number of tokens in topic statement including stop words.

	Min	Max	Mean		Min	Max	Mean
TREC-1 (51-100)	44	250	107.4	TREC-5 (251-300)	29	213	82.7
title	1	11	3.8	title	2	10	3.8
description	5	41	17.9	description	6	40	15.7
narrative	23	209	64.5	narrative	19	168	63.2
concepts	4	111	21.2				
TREC-2 (101-150)	54	231	130.8	TREC-6 (301-350)	47	156	88.4
title	2	9	4.9	title	1	5	2.7
description	6	41	18.7	description	5	62	20.4
narrative	27	165	78.8	narrative	17	142	65.3
concepts	3	88	28.5				
TREC-3 (151-200)	49	180	103.4	TREC-7 (351-400)	31	114	57.6
title	2	20	6.5	title	1	3	2.5
description	9	42	22.3	description	5	34	14.3
narrative	26	146	74.6	narrative	14	92	40.8
TREC-4 (201-250)	8	33	16.3	TREC-8 (401-450)	23	98	51.8
				title	1	4	2.5
				description	5	32	13.8
				narrative	14	75	35.5

```

<num> Number: 409
<title> legal, Pan Am, 103

<desc> Description:
What legal actions have resulted from the destruction
of Pan Am Flight 103 over Lockerbie, Scotland, on
December 21, 1988?
<narr> Narrative:
Documents describing any charges, claims, or fines
presented to or imposed by any court or tribunal are
relevant, but documents that discuss charges made in
diplomatic jousting are not relevant.

```

Figure 2: A sample TREC-8 topic.

in both collections and balancing the load across assessors.

3.3 Relevance assessments

Relevance judgments are of critical importance to a test collection. For each topic it is necessary to compile a list of relevant documents—as comprehensive a list as possible. All TRECs have used the pooling method [9] to assemble the relevance assessments. In this method a pool of possible relevant documents is created by taking a sample of documents selected by the various participating systems. This pool is then shown to the human assessor, who makes a binary (yes/no) relevance judgment for each document in the pool. Unjudged documents are assumed to be not relevant. The particular sampling method used in TREC is to take the top 100 documents retrieved per judged run for a given topic and merge them into the pool for assessment. This is a valid sampling technique since all the systems used ranked retrieval methods, with those documents

Table 4: Overlap of submitted results

	Possible	Actual	Relevant
TREC-1	3300	1279 (39 %)	277 (22 %)
TREC-2	4000	1106 (28 %)	210 (19 %)
TREC-3	2700	1005 (37 %)	146 (15 %)
TREC-4	7300	1711 (24 %)	130 (08 %)
ad hoc	4000	1345	115
confusion	900	205	0
dbmerge	800	77	2
interactive	1600	84	13
TREC-5	10,100	2671 (27 %)	110 (04 %)
ad hoc	7700	2310	104
dbmerge	600	72	2
NLP	1800	289	3
TREC-6	3,430	1445 (42 %)	92 (06 %)
ad hoc	3100	1326	89
NLP	200	113	2
HP	130	6	1
TREC-7	7,805	1611 (21 %)	93 (06 %)
ad hoc	7700	1605	92
HP	105	6	.5
TREC-8	7,100	1736 (25 %)	94 (05 %)

most likely to be relevant returned first. Each pool is sorted by document identifier so assessors cannot tell if a document was highly ranked by some system or how many systems (or which systems) retrieved the document.

To keep the assessment task manageable, only a subset of the runs that are submitted to NIST are judged (that is, contribute to the assessment pools). When participants submit their runs, they rank the submissions in the order they prefer them to be judged. NIST ensures that the same number of runs from each participant is used when creating the pools (provided the participant has submitted that many runs), and uses the top 100 documents for every topic from every judged run. This strategy does not take advantage of recent proposals such as those by Zobel [14] or Cormack, Palmer, and Clarke [2] for finding more relevant documents in fewer total documents judged. Besides being difficult to handle logistically at NIST, there are concerns about how implementing these proposals might bias the assessments. Zobel suggests judging more documents for topics that have had many relevant documents found so far and fewer documents for topics with fewer relevant documents found so far as a way to improve the completeness of the pools. However, assessors would know that documents added later in the pools came from lower in the systems' rankings and that may affect their judgments. Cormack et al. suggest judging more documents from runs that have returned more relevant documents recently and fewer documents from runs that have returned fewer relevant document recently. But that would bias the pools towards systems that retrieve relevant documents early in their rankings. For test collections, a lack of bias in the relevance judgments is more important than the total number of relevant documents found.

3.3.1 Overlap

Table 4 summarizes the amount of overlap in the ad hoc pool for each TREC. The first data column in the table gives the maximum possible size of the pool. Since the top 100 documents from each run are judged, this number is usually 100 times the number of runs used to form the pool, though in some years track runs contributed fewer than 100 documents. The next column shows the number of documents that were actually in the pool (i.e., the number of unique documents retrieved in the top 100 across all judged runs) averaged over the number of topics. The percentage given in that column is the size of the actual pool relative to the

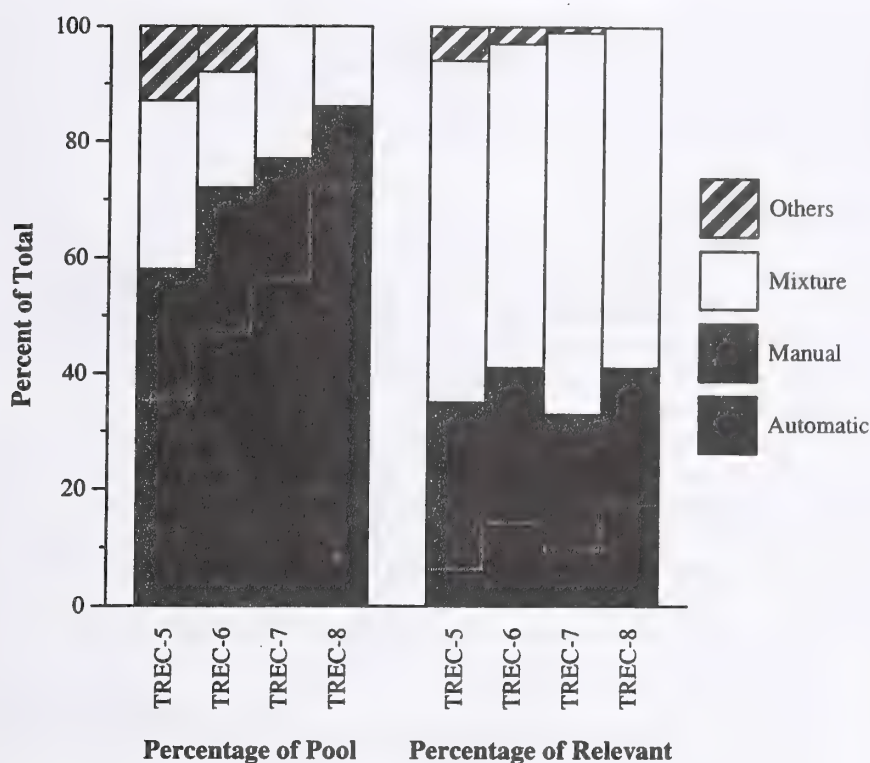


Figure 3: Mean percentage of pools and relevant documents by category. Means are computed over the 50 topics contained in the test sets.

possible pool size. The final column gives the average number of relevant documents in the pool and the percentage of the actual pool that was relevant. For TRECs 4–7, various tracks also contributed documents to the ad hoc pool. These are broken out in the appropriate rows within table 4, where the order of the tracks is significant (a document retrieved in a track listed later is not counted for that track if the document was also retrieved by a track listed earlier). The TREC-8 ad hoc pools were created only from ad hoc runs.

The average overlap found in the pools has been stable since TREC-4 except for TREC-6. The tremendous drop in the size of the ad hoc pool for that year reflects the difference in the number of runs NIST was able to assess that year. Table 4 also shows that the average number of relevant documents per topic has remained stable after decreasing from an early high. NIST has deliberately chosen more tightly focused topics to better guarantee the completeness of the relevance assessments.

The figures for average overlap given in table 4 hide details about the source of the documents in the pool. Figure 3 shows the mean percentage of the total pool and the mean percentage of the total number of relevant documents contributed by each type of ad hoc run for TRECs 5–8. In the figure, “Automatic” designates documents that were retrieved only by automatic runs, “Manual” designates documents that were retrieved only by manual runs, “Mixture” designates documents that were retrieved by runs of different types, and “Others” designates documents that were retrieved by other tracks that contributed to the ad hoc pools. For example, for TREC-8 72 % of the pool came from the automatic runs, 14 % of the pool from manual runs, and 14 % of the pool came from runs of both types. In contrast, 17 % of the relevant documents came from automatic runs, 24 % of the relevant documents came from manual runs, and 59 % of the relevant documents came from runs of both types. For each of the years shown, the majority of the relevant documents were retrieved by multiple categories of runs. Manual runs retrieved a higher percentage of the relevant documents than they contributed to the pools.

Figure 4 gives a different view of the same issue by looking at the groups that retrieved unique relevant documents—relevant documents that were contributed to the pool by exactly one group. The figure contains a histogram of the total number of unique relevant documents found by a group over the 50 test topics for each of the last four TRECs. The totals are subdivided using the same categories as were used in figure 3.

Each of the histograms in the figure uses the same scale. A dot underneath the x-axis indicates a group is plotted there, and all groups that retrieved at least one unique relevant document are plotted. For each year, the majority of unique documents was retrieved by manual runs. The distribution of unique relevant documents found has been roughly the same over the four years.

3.3.2 Effect of pooling on evaluation

Some people object to the use of pooling to produce a test collection because unjudged documents are assumed to be not relevant. They argue that evaluation scores for methods that did not contribute to the pools will be deflated relative to methods that did contribute because the non-contributors will have highly ranked unjudged documents. Displays such as figure 4 contribute to these fears since they demonstrate that some relevant documents are found by only one group. If that group had not participated, those relevant documents would not have been judged.

Zobel demonstrated that the quality of the pools (the number and diversity of runs contributing to the pools and the depth to which those runs are judged) does affect the quality of the final collection [14]. He also found that the TREC collections were not biased against unjudged runs. In this test, he evaluated each run that contributed to the pools using both the official set of relevant documents published for that collection and the set of relevant documents produced by removing the relevant documents uniquely retrieved by the run being evaluated. For the TREC-5 ad hoc collection, he found that using the unique relevant documents increased a run's 11 point average precision score by an average of $\frac{1}{2}$ %. The maximum increase for any run was 3.5 %. The average increase for the TREC-3 ad hoc collection was somewhat higher at 2.2 %.

We can perform a similar check of a test collection as soon as relevance judging is complete. For each run that contributed to the pool, we compute the mean average precision (see section 4 for a definition of mean average precision) of the run using the standard relevance judgments and the set of relevance judgments produced by removing the relevant documents uniquely retrieved by that run's group. The mean percentage difference in mean average precision over the 71 runs that contributed to the ad hoc pool was 0.78 %, with a maximum difference of 9.9 %. Not surprisingly, the manual groups that had the largest number of unique relevant documents (see figure 4) also had the largest percentage differences in mean average precision. But given that the manual runs' contributions are in the pool, the difference in evaluation results for automatic runs is negligible. For automatic runs, the largest percentage difference in mean average precision scores was 3.85 %, which corresponded to an absolute difference of only .0001. Every automatic run that had a mean average precision score of at least .1 had a percentage difference of less than 1 %.

Figure 5 shows the absolute difference in mean average precision scores plotted against the number of unique relevant documents contributed by that run's group for each automatic run. The runs are sorted by increasing difference and then by number of unique relevant documents. The two obvious outliers in number of unique relevant documents (for runs GE8ATDN1 and iit99au1) reflect organizations that submitted manual runs in addition to automatic runs; the vast majority of their unique relevant documents were contributed by their manual run.

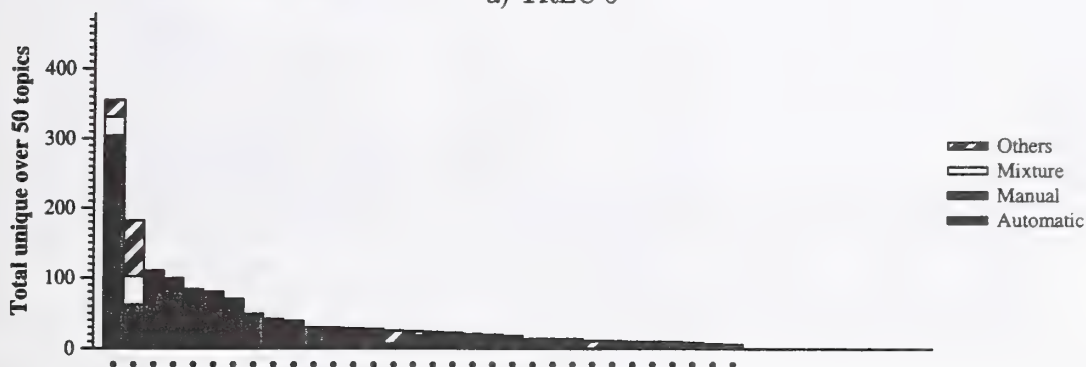
While the lack of any appreciable difference in the scores of the automatic runs is not a guarantee that all relevant documents have been found, it is very strong evidence that the test collection is reliable for comparative evaluations of retrieval runs. Note that differences of less than 1 % are smaller than the differences that result from using different relevance assessors [12]. The quality of the pools is significantly enhanced by the presence of the recall-oriented manual runs. The organizers of the NTCIR (NACSIS Test Collection for evaluation of Information Retrieval systems) workshop took good advantage of this effect by performing their own manual runs to supplement their pools [6].

4 Evaluation

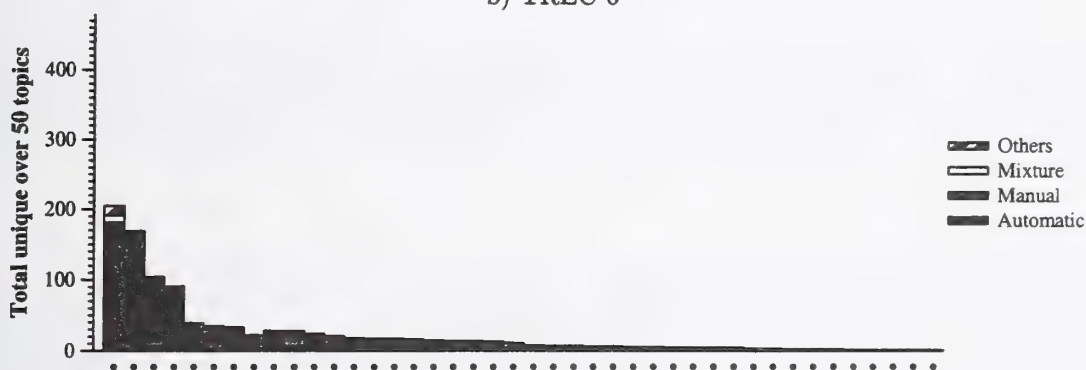
The entire purpose of building a test collection is to be able to compare the effectiveness of retrieval systems. Providing a common evaluation scheme is an important element of TREC.



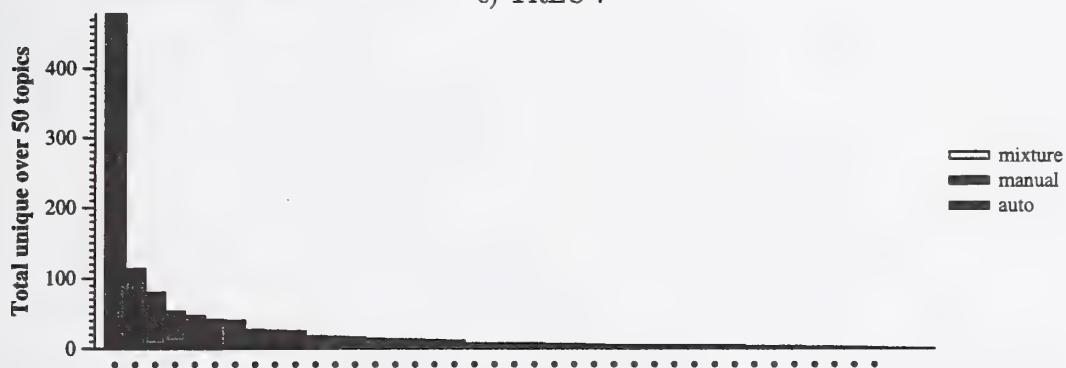
a) TREC-5



b) TREC-6



c) TREC-7



d) TREC-8

Figure 4: Total number of unique relevant documents retrieved per TREC. Each total gives the percentages of the total that were retrieved by Automatic, Manual, Mixed, or Other runs. Groups are indicated by a dot beneath the x-axis. All groups that retrieved at least one unique relevant document are plotted.

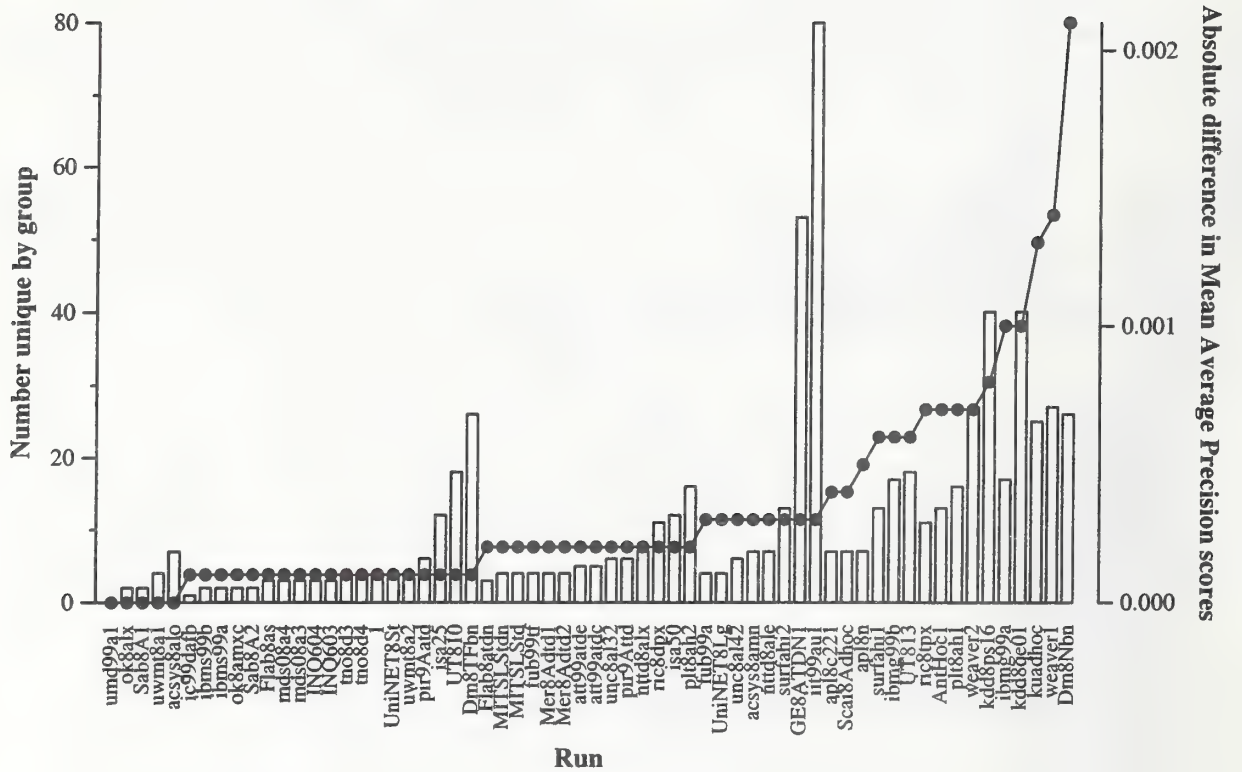


Figure 5: Absolute difference in mean average precision scores when a run is evaluated using relevance pools with and without that group’s unique relevant document for TREC-8 automatic, ad hoc runs. Also plotted is the number of unique relevant documents contributed to the pools by that group. Runs are ordered by increasing absolute difference and by increasing number of unique relevant documents.

4.1 Current practice

All TREC tasks that involve returning a ranked list of documents are evaluated using the `trec_eval` package. This package, written by Chris Buckley, reports about 85 different numbers for a run. The measures reported include *recall* and *precision* at various cut-off levels plus single-valued summary measures that are derived from recall and precision. Precision is the proportion of retrieved documents that are relevant, while recall is the proportion of relevant documents that are retrieved. A cut-off level is a rank that defines the retrieved set; for example, a cut-off level of ten defines the retrieved set as the top ten documents in the ranked list. The `trec_eval` program reports the scores as averages over the set of topics where each topic is equally weighted. (The alternative is to weight each relevant document equally and thus give more weight to topics with more relevant documents. Evaluation of retrieval effectiveness historically weights topics equally since all users are assumed to be equally important.)

Precision reaches its maximal value of 1.0 when only relevant documents are retrieved, and recall reaches its maximal value (also 1.0) when all the relevant documents are retrieved. Note, however, that these theoretical maximum values are not obtainable as an average over a set of topics at a single cut-off level because different topics have different numbers of relevant documents. For example, a topic that has fewer than ten relevant documents will have a precision score less than one after ten documents are retrieved regardless of how the documents are ranked. Similarly, a topic with more than ten relevant documents must have a recall score less than one after ten documents are retrieved. At a single cut-off level, recall and precision reflect the same information, namely the number of relevant documents retrieved. At varying cut-off levels, recall and precision tend to be inversely related since retrieving more documents will usually increase recall while degrading precision and vice versa.

This overview paper generally uses two evaluation measures when discussing retrieval results, the recall-precision curve and mean (non-interpolated) average precision. A recall-precision curve plots precision as a function of recall. Since the actual recall values obtained for a topic depend on the number of relevant documents, the average recall-precision curve for a set of topics must be interpolated to a set of standard recall values. The particular interpolation method used is given in Appendix A, which also defines many of the other evaluation measures reported by `trec_eval`. Recall-precision graphs show the behavior of a retrieval run over the entire recall spectrum.

Mean average precision is the single-valued summary measure used when an entire graph is too cumbersome. The average precision for a single topic is the mean of the precision obtained after each relevant document is retrieved (using zero as the precision for relevant documents that are not retrieved). The mean average precision for a run consisting of multiple topics is the mean of the average precision scores of each of the individual topics in the run. The average precision measure has a recall component in that it reflects the performance of a retrieval run across all relevant documents, and a precision component in that it weights documents retrieved earlier more heavily than documents retrieved later. Geometrically, mean average precision is the area underneath a non-interpolated recall-precision curve.

The (reformatted) output of `trec_eval` for each submitted run is given in Appendix A. In addition to the ranked results, participants are also asked to submit data that describes their system features and timing figures to allow a primitive comparison of the amount of effort needed to produce the corresponding retrieval results. These system descriptions are not included in the printed version of the proceedings due to their size, but they are available on the TREC web site (<http://trec.nist.gov>).

5 Ad Hoc Retrieval Results

This section briefly summarizes some of the approaches used for the ad hoc task. To recap the specific task that was to be performed, the TREC-8 ad hoc task entailed using new topics 401–450 to search the documents on Disks 4 and 5 minus the *Congressional Record* documents. A run was either automatic or manual. For an automatic run, all processing was done by the machine without human intervention of any sort. All other runs were manual runs. There were 129 ad hoc runs submitted for the task: 13 manual runs and 116 automatic runs.

5.1 Automatic results

Of the 116 automatic runs submitted for the ad hoc task, 37 runs used the complete topic statement, 59 runs used only the title and description fields, and 20 used only the title field. A run that used only the title and description fields (so-called “short” runs) was required from every group that submitted any automatic run. Figure 6 shows the recall/precision curves for the eight TREC-8 groups with the best short runs as measured by mean average precision. The runs are ranked by average precision and only one run is shown per group. These graphs (and others in this section) are not intended to show specific comparison of results across sites but rather to provide a focal point for discussion of methodologies used in TREC. For more details on the various runs and procedures, please see the cited papers in this proceedings.

pir9Attd – Queens College, CUNY (“TREC-8 Ad-Hoc, Query and Filtering Track Experiments using PIRCS” by K.L. Kwok, L. Grunfeld, and M. Chan). The PIRCS system is a spreading activation method, which the authors show can be viewed as a combination of a probabilistic model and a simple language model. This run was produced using the same basic processing as the CUNY TREC-7 runs. The final result is produced by using a sequence of 5 different techniques that improve on the initial result. These methods include average within-document term frequency weights for query terms, a variable Zipf threshold for selecting indexing terms, collection enrichment (using documents from outside the collection in the first stage retrieval to improve the density of relevant documents), query expansion by adding highly associated terms based on a mutual information measure, and reweighting query terms based on the retrieved set.

ok8amxc – Okapi group (“Okapi/Keenbow at TREC-8” by S.E. Robertson and S. Walker). This run used the BM25 weighting scheme developed several years ago by this group and query expansion based on

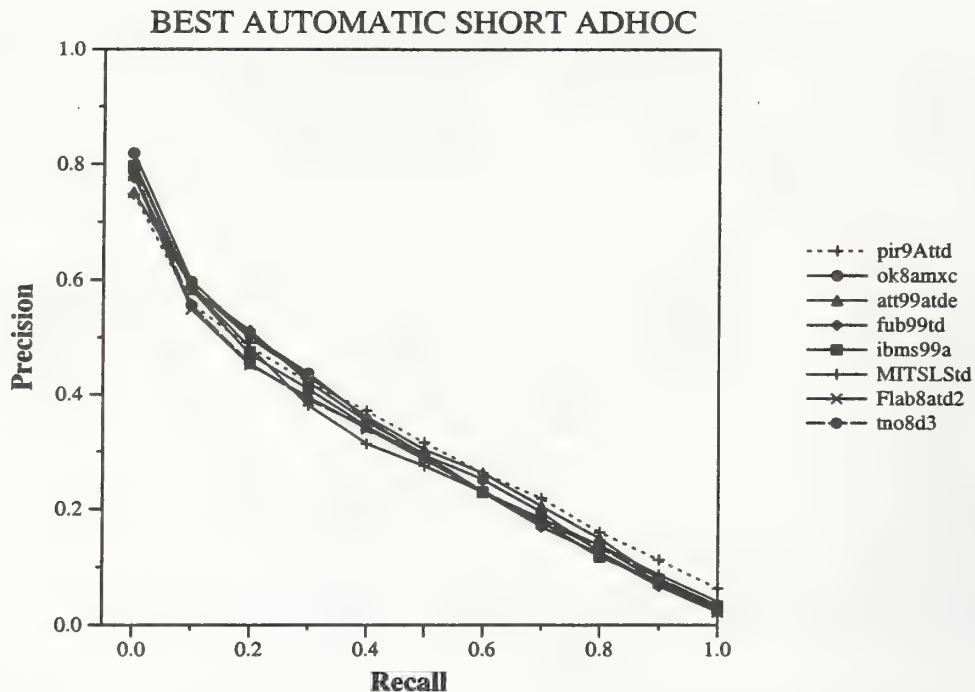


Figure 6: Recall/Precision graph for the top eight automatic short ad hoc runs.

blind feedback. The process used to select which terms should be included in the expanded query was new. Instead of simply taking the X candidate terms with the highest relevance weights, terms were added to the query if their relevance weight exceeded an absolute threshold.

att99atde – AT&T Labs–Research (“AT&T at TREC-8” by A. Singhal, S. Abney, M. Bacchiani, M. Collins, D. Hindle, and F. Pereira). AT&T also used essentially the same processing for TREC-8 as they used in TREC-7. This processing is based on a vector-space model with length-normalized, $tf \times idf$ weights, and query expansion using blind feedback and conservative collection enrichment. For this run, the blind feedback step was tweaked slightly. The initial retrieval pass produced a ranking of the top 50 documents, which was then re-ranked by promoting documents that contain multiple query terms. The top 10 documents from the new ranking were assumed to be relevant for feedback.

fub99td – Fondazione Ugo Bordoni (“TREC-8 Automatic Ad-Hoc Experiments at Fondazione Ugo Bordoni” by C. Carpineto and G. Romano). This run was produced using the Okapi formula for retrieving an initial set of documents and then expanding queries based on an information-theoretic term scoring function. The term scoring function uses the difference of the distribution of a term in the presumed-relevant set and the entire collection to compute the score. The same function was used in this group’s TREC-7 work, but the ranking in the initial set of documents was much improved this year leading to much better overall retrieval.

ibms99a – IBM T.J. Watson Research Center (“Ad hoc, Cross-language and Spoken Document Information Retrieval at IBM” by M. Franz, J.S. McCarley, and R.T. Ward). This run was produced using the same processing as the IBM group’s submissions to earlier TRECs. Okapi weights are used to provide an initial ranking of passages. Queries are then expanded using an LCA-like procedure based on top-ranking passages. The final document ranking is created using Okapi weights and the newly expanded query, with the score for a document computed as a function of the document’s score and the score of its highest ranking passage.

MITSLStd – MIT Laboratory for Computer Science (“A Maximum Likelihood Ratio Information Retrieval Model” by K. Ng). The MIT group introduced a new probabilistic model based on the change in

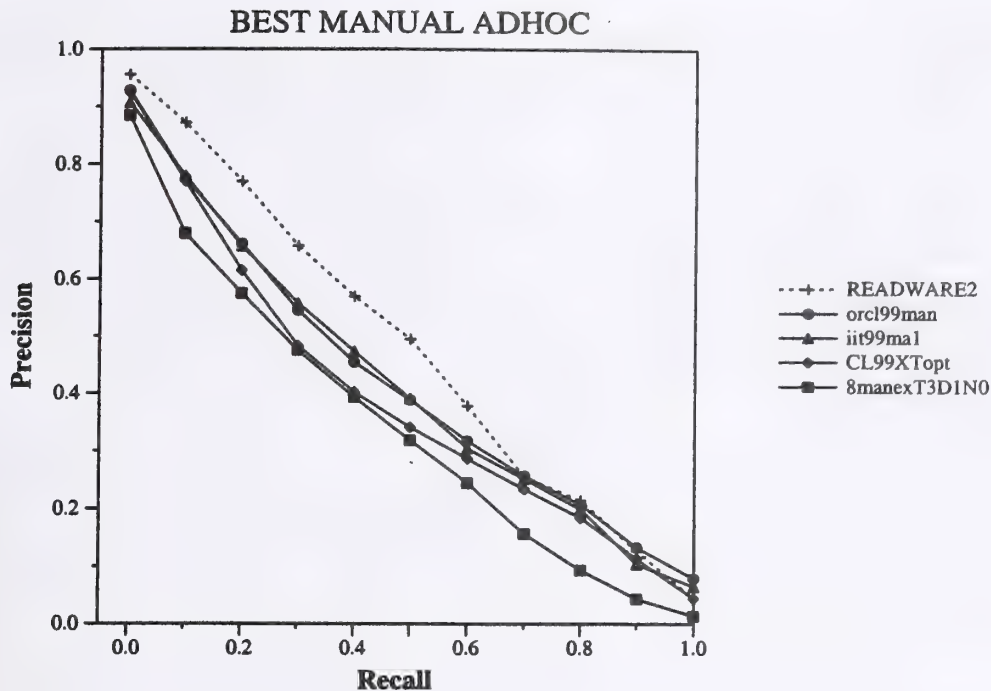


Figure 7: Recall/Precision graph for the top five manual ad hoc runs.

the likelihood of a document once the query is issued as compared to its a priori probability. The likelihoods are estimated using statistical language modeling techniques. This run was produced using the new model with an extension to incorporate blind feedback.

Flab8atd2 – Fujitsu Laboratories (“Fujitsu Laboratories TREC8 Report—Ad hoc, Small Web, and Large Web Track” by I. Namba and N. Igata). Fujitsu Laboratories experimented with a number of techniques, many of which were either unstable (i.e., significantly improving some topics while significantly degrading an equal number of topics) or provided little benefit. This run used a modified Okapi weighting scheme, incorporated query expansion using blind feedback, increased the score of documents that contained multiple query terms, and increased the score of documents that contained good word pairs from the topic.

tno8d3 – Twenty-One project (“Twenty-One at TREC-8: using Language Technology for Information Retrieval” by W. Kraaij, R. Pohlmann, and D. Hiemstra). The Twenty-One project uses a vector-space model where weights are based on statistical language models. Their TREC-8 processing was the same as their TREC-7 processing, which includes query expansion using blind feedback.

The immediate conclusion to be drawn from figure 6 is that there are many approaches that lead to essentially the same retrieval effectiveness. Yet while there are differences in the details of these approaches, they all share two properties that we can therefore conclude are fundamental to effective retrieval performance. Of primary importance is the use of a high-quality weighting scheme. Query expansion using terms from highly-ranked documents or documents related to highly-ranked documents is also beneficial.

5.2 Manual results

Figure 7 shows the recall/precision curves for the five TREC-8 groups with the highest mean average precision scores for manual runs. Once again, the runs are ranked by mean average precision and only one run per group is shown.

READWARE2 – Management Information Technologies, Inc. (“High Selectivity and Accuracy with READWARE’s Automated System of Knowledge Organization” by T. Adi, O.K. Ewell, and P. Adi). This

run was produced by an analyst using READWARE's tools to define and refine a set of highly-specific queries for each topic. The number of queries used per topic ranged between 2 and 65 with a mean of 14. The submitted results were the union of the output of the different queries.

orcl99man – Oracle Corporation (“Oracle at Trec8: a Lexical Approach” by K. Mahesh, J. Kud, and P. Dixon). Oracle's interMedia Text retrieval system includes a large lexical knowledge base that provides a hierarchical classification of concepts with cross-reference links among concepts. The system provides an “ABOUT” operator that allows queries that specify a high-level concept to match documents that contain a particular expression of that concept. This run was produced by having a searcher interact with the interMedia system (using manual feedback, browsing the lexical knowledge base, etc.) to define a single (assumed-to-be-best) query for each topic. The submitted results were the ranked output produced by the final query.

iit99ma1 – Illinois Institute of Technology group (“IIT at TREC-8: Improving Baseline Precision” by M.C. McCabe, D.O. Holmes, K.L. Alford, A. Chowdhury, D.A. Grossman, and O. Frieder). This run is the latest in a series of manual runs the IIT group has submitted to the past several TRECs. For this run, the searcher spent approximately a half hour formulating a query using manual relevance feedback and general knowledge to select query words. The basic retrieval strategy was a vector-space model, though there was also a mechanism to remove a document from the retrieved set if it contained concepts that were included on a (topic-specific) negated concepts list.

CL99XTopt – CLARITECH Corporation (“CLARIT TREC-8 Manual Ad-Hoc Experiments” by D.A. Evans, J. Bennett, X. Tong, A. Huettnner, C. Zhai, and E. Stoica). In previous years, the CLARITECH group showed that clustering the result set of a search helped users find relevant documents to use in a subsequent round of relevance feedback. This year, the system enabled the clustering to be computed over an extended set of index terms. A second change allowed the number of query terms added to the query during relevance feedback to be query-dependent rather than an arbitrary, fixed number. Searchers were allowed a maximum of 20 minutes wall clock time per topic. In addition to making relevance judgments, the searchers could modify the automatically-constructed query if they chose to do so. This run was the result of both the new clustering and the query-dependent number of expansion terms.

8manexT3D1N0 – GE Research and Development group (“Natural Language Information Retrieval: TREC-8 Report” by T. Strzalkowski, J. Perez-Carballo, J. Karlgren, A. Hulth, P. Tapanainen, and T. Lahtinen). This run was the result of an investigation into how effective natural language indexing techniques are when query statements are large. The original TREC topic statement was fed to a standard retrieval system and topic-related summaries of the top 30 documents were returned. The user reviewed each summary and removed any summary that was not relevant. This was the only manual processing in the run, and users were limited to no more than 10 minutes wall-clock time to perform the review. All summaries not explicitly removed by the user were attached to the original topic statement, and the resulting new statement was submitted to the groups' NLP-based retrieval system.

Comparisons of manual runs in the ad hoc task are especially tricky because a wide range of levels of human effort are lumped together into a single category. Comparing the graphs in figures 6 and 7, at a minimum we can conclude that users who will participate in their searches can be rewarded with better search results.

5.3 Future of the ad hoc main task

The ad hoc task was one of the two tasks that were performed in TREC-1 and has been run in every TREC since then. There are several reasons for the task's primary position in TREC.

- Historically, ad hoc retrieval has been regarded as the fundamental task in text retrieval.
- Having one task that all (or most) groups perform documents the state of the art and provides a basis of comparison for each groups' results in other tracks.

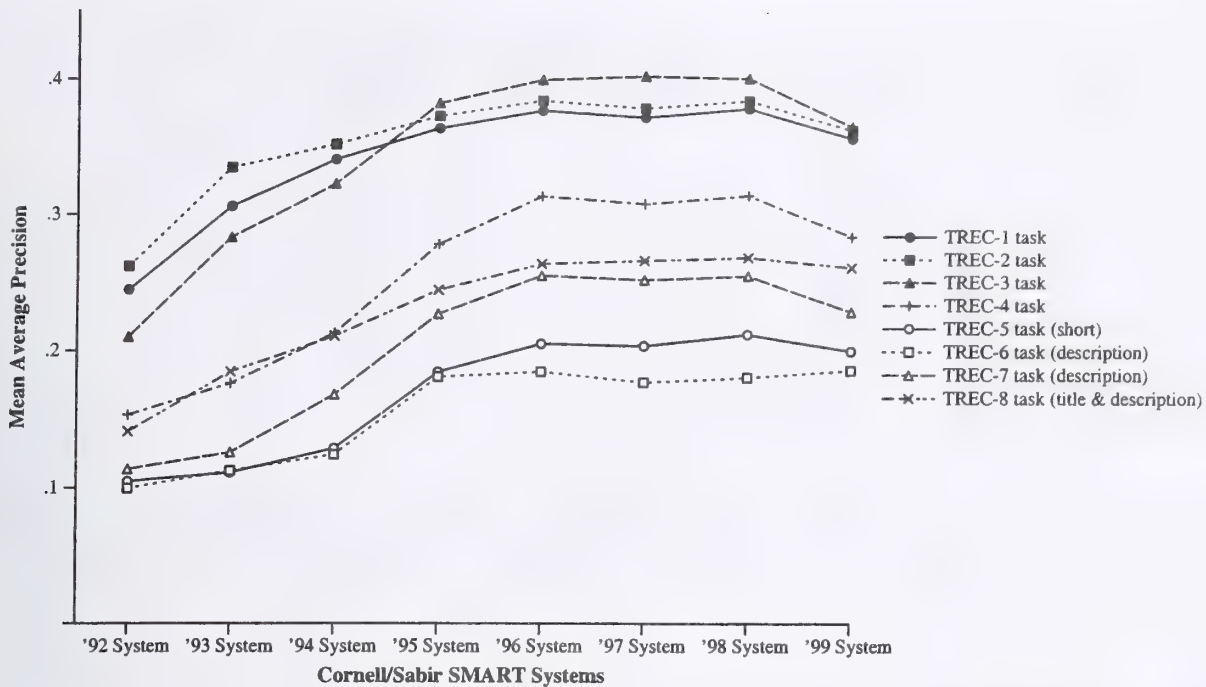


Figure 8: Mean average precision obtained by different versions of the Cornell/Sabir SMART system for the different ad hoc tasks in TREC.

- The main task is the means by which the TREC general-purpose IR test collections are built.
- The task provides a convenient starting point for new participants.

However, a number of participants also believe that the community is no longer learning enough from the results of the task to justify the (participants' and NIST) resources spent on it. Many participants' TREC-8 runs were produced using essentially the same system as previous years' runs simply to satisfy the requirement for an ad hoc run.

Figure 8 shows how the results in the ad hoc task have plateaued in recent years. The figure gives a plot of the mean average precision scores obtained by the Cornell/Sabir SMART system for each version of the system for each ad hoc task (i.e., topic and document sets) as reported by the SMART group [1]. Each line connects the scores for the same task across the system variants. The '92 System was the system that was used to produce the runs submitted to TREC-1, the '93 System was used to produce the runs submitted to TREC-2, etc. Once a new set of topics was released, the SMART group ran that set of topics on each prior version of the system. The SMART system has consistently been among the better systems in TREC (except the '99 System in which a technique designed to enhance early precision turned out to harm overall performance), so the trend is indicative of the field as a whole. Retrieval effectiveness in the ad hoc task has improved dramatically since the beginning of TREC but has now leveled off.

Because of these considerations and the fact that we now have 8 years worth of test collections, the ad hoc main task will be discontinued in future TRECs. This is not to say that we believe that the ad hoc text retrieval problem is solved. Indeed, figure 8 shows absolute performance on the task is less than ideal. Rather it is an acknowledgement that sufficient infrastructure exists so that researchers can pursue their investigations independently, and thereby free TREC resources for other tasks.

6 The Tracks

One of the goals of TREC is to provide a common task evaluation that allows cross-system comparisons, which has proven to be a key strength in TREC. A second major strength is the loose definition of the ad hoc

Table 5: Number of task participants.

	TREC-6	TREC-7	TREC-8
Ad Hoc	31	42	41
CLIR	13	9	12
Filtering	10	12	14
GIRT	—	0	2
Interactive	9	8	7
QA	—	—	20
Query	—	2	5
SDR	13	10	10
Small Web	—	—	17
Large Web	—	—	8

task, which allows a wide range of experiments. The addition of secondary tasks (called tracks) in TREC-4 combined these strengths by creating a common evaluation for retrieval subproblems.

The tracks have had a significant impact on TREC participation. Table 5 gives the number of participants in each of the TREC-8 tasks for TREC 6, 7, and 8. The total number of participating groups continues to grow each year, with 66 groups this year compared to 56 in TREC-7 and 51 in TREC-6.

Each track has a set of guidelines developed under the direction of the track coordinator. Participants are free to choose which, if any, of the tracks they will join. This section describes the tasks performed in TREC-8 tracks. See the track reports elsewhere in this proceedings for a more complete description of each track.

6.1 The Cross-Language (CLIR) track

The CLIR task focuses on retrieving documents that are written in different languages using topics that are in one language. The TREC-8 track used the same document set that was used in TREC-7: a set of French documents from the Swiss news agency *Schweizerische Depeschen Agentur* (SDA); a set of German documents from SDA plus a set of articles from the newspaper *New Zurich Newspaper* (NZZ); a set of Italian documents from SDA; and a set of English documents from the AP newswire. All of the document sets contain news stories from approximately the same time period, but are not aligned or specially coordinated with one another. Participants were provided with a new set of 28 topics, numbered 54 through 81, that had translations available in English, French, German, and Italian. Participants used one topic language to search the combined document set.

As in TREC-7, the construction of the cross-language test collection differed from the way other TREC collections are created. Candidate topics in the native language were created in each of four different institutions: NIST, USA (English); University of Zurich, Switzerland (French); Social Science Information Centre, Bonn/University of Koblenz, Germany (German); and CNR, Pisa, Italy (Italian). Each institution developed candidate topics such that a third of the candidates targeted international events, a third targeted items of interest in Europe generally, and a third targeted local items of interest. The intention was to create topics that had different distributions of relevant documents across languages. Each of the institutions contributed seven topics to the final set of 28, with representatives from each site meeting to ensure the actual question being asked in the topic was understood by all. The final topics were then translated into the three remaining languages so that the entire set of topics was available in each language. The relevance judgments for all topics for a particular document language were made at the site responsible for that language. The TREC-7 and TREC-8 cross-language collections are the only TREC collections in which multiple relevance assessors provided judgments for a single topic.

Forty-five runs from 12 different groups were submitted to the track. Eight runs were submitted for the special GIRT task (described below) and nine runs used a subset of the document collection (for example, Italian topics run against the English portion of the collection). Only one run was a manual run.

Evaluation of cross-language retrieval poses some challenges. As mentioned above, this is the only task

in which multiple assessors judge the same topic. Pool creation is also affected: pools must be adequately balanced across languages to assure sufficient coverage in each language. For TREC-8, 18 runs were added to the pools. These runs included each group's first choice of runs to be judged plus runs that retrieved relatively many German or Italian documents. In addition, a monolingual Italian run was solicited by NIST to bolster the Italian pools.

We can perform the same analysis as was performed for the ad hoc collection in section 3.3.2 once the pools are judged. For the cross-language runs, the mean percentage difference in mean average precision scores computed with and without a group's unique relevant documents was 6.3 %, with a maximum percentage difference of 15.4 %. This compares with the ad hoc maximum percentage difference of less than 1 % for automatic runs. The larger difference for the cross-language test collection can be attributed to the combination of fewer runs contributing to the pools and the lack of high-recall manual runs. While the difference is clearly larger for the cross-language test collection than the ad hoc collection, a mean difference of approximately 6 % is okay for most purposes. Experimenters who find many unjudged documents in the top-ranked list of only one of a pair of runs to be contrasted may need to proceed with care.

The TREC-8 track also had an optional subtask known as the **GIRT task**. The subtask used the GIRT collection, a 31,000 document structured database (formatted as SGML fielded text data) from the field of social science, plus the NZZ articles, and a separate set of 28 topics. The rationale of the subtask was to study CLIR in a vertical domain (i.e., social science) where a German/English thesaurus is available.

6.2 The Filtering track

The tasks within the TREC-8 filtering track were the same as the TREC-7 track, though the document and topic sets differed and as did the utility functions used to evaluate the runs. The TREC-8 track used topics 351–400 and the *Financial Times* document set from Disk 4.

The filtering problem can be viewed as the inverse of the ad hoc retrieval task in that the question is assumed to be known and the document stream changes. The filtering task is to retrieve just those documents in the stream that match the user's interest as represented by the query. The main focus of the track was an *adaptive* filtering task. In this task, a filtering system starts with just a query derived from the topic statement, and processes documents one at a time in date order. If the system decides to retrieve a document, it obtains the relevance judgment for it, and can modify the query based on the judgment if desired.

For continuity with previous TRECs, two other, simpler tasks were also part of the TREC-8 track. In the *batch* filtering task, the system is given a topic and a set of known relevant documents. The system creates a query from the topic and known relevant documents, and must then decide whether or not to retrieve each document in the test portion of the collection. In the *routing* task, the system again builds a query from a topic statement and a set of relevant documents, but then uses the query to rank the test portion of the collection. Ranking the collection by similarity to the query (routing) is an easier problem than making a binary decision as to whether a document should be retrieved (batch filtering) because the latter requires a threshold that is difficult to set appropriately.

Fifty-five runs from 14 different groups were submitted to the filtering track. Thirty-three of the runs were adaptive filtering runs, 11 runs were batch filtering runs, and 11 runs were routing runs. The results of the adaptive filtering subtask demonstrate the difficulty of the problem. When evaluated using the rule that retrieving a relevant document earns a system three "points" while retrieving a nonrelevant document subtracts two "points" (the LF1 utility function below), the average behavior of each system was worse than the baseline of retrieving no documents at all. With a somewhat easier scoring metric (three points for a relevant retrieved but only 1 point subtracted for a nonrelevant retrieved) some systems performed better than the baseline on average, but very small retrieved sets were still best.

Developing appropriate measures for filtering systems continues to be an important part of the track. The main approach used in TREC is to use utility functions as measures of the quality of the retrieved set—the quality is computed as a function of the benefit of retrieving a relevant document and the cost of retrieving an irrelevant document [7]. In TREC-8 two different linear utility functions were used:

$$\text{LF1} = 3R^+ - 2N^+$$

$$\text{LF2} = 3R^+ - N^+$$

where R^+ and N^+ are the number of relevant and non-relevant documents retrieved, respectively. A pair of non-linear utility functions were also defined for the TREC-8 task, but few runs that were optimized for those metrics were submitted. Many participants in the track felt that the non-linear measures did not model a user's behavior very well.

6.3 The Interactive track

The interactive track was one of the first tracks to be introduced into TREC. Since its inception, the high-level goal of the track has been the investigation of searching as an interactive task by examining the process as well as the outcome. One of the main problems with studying interactive behavior of retrieval systems is that both searchers and topics generally have a much larger effect on search results than does the retrieval system used.

The TREC-8 task was very similar to the TREC-7 task. The track used slightly modified versions of six ad hoc topics. Each of the six topics described an information need such that the document collection (the *Financial Times* collection from Disk 4) contained multiple distinct examples or instances of the requested information. The searchers' job was to save documents covering as many distinct answers to the question as possible in a 20-minute time limit. The NIST assessor for the topic made a comprehensive list of instances from the documents submitted by the track. The effectiveness of the search was evaluated by the fraction of total instances for that topic covered by the search (instance recall) and the fraction of the documents retrieved in the search that contained an instance (instance precision). Participants were also required to collect demographic and user satisfaction data from the searchers, and to report extensive data on each searcher's interactions with the search systems.

The track did not attempt to coordinate cross-site comparisons or test a particular hypothesis across sites. It did impose an experimental matrix that defined how searchers and topics were to be divided among whatever experimental and control systems the participants were testing. The matrix was based on a latin square design to provide an uncontaminated estimate of the difference between the systems. The minimum experiment defined by the design required 12 searchers so that query order would not be confounded with other effects. Each searcher performed three searches with each of the two systems, and each query was searched in each position (first through sixth) by each system.

Seven groups submitted interactive results. Two groups used the minimum experimental design of 12 searchers, four groups used 24 searchers, and one group used 36 searchers. Each group found little difference between their control and experimental systems. This could mean that none of the various devices implemented in the experimental systems are helpful in the instance retrieval task, or that the statistical power of the experimental design is not sufficient to detect the difference. Further study of the design of effective user studies is needed.

6.4 The Question Answering track

TREC-8 was the first time the question answering track was run. The purpose of the track was to encourage research into systems that return actual answers, as opposed to ranked lists of documents, in response to a question.

The track used the ad hoc document collection (i.e., the documents on Disks 4 and 5 minus the *Congressional Record* subcollection) and 198 fact-based, short-answer questions such as "How many calories are there in a Big Mac?" Each question was guaranteed to have at least one document in the collection that answered the question. Participants were to return a ranked list of five strings per question such that each string was believed to contain an answer to the question. Depending on the run type, answer strings were limited to either 50 or 250 bytes. Human assessors read each string and made a binary decision as to whether or not the string actually did contain an answer to the question. Individual questions received a score equal to the reciprocal of the rank at which the first correct response was returned (or 0 if none of the five responses contained a correct answer). The score for a run was the mean of the individual questions' reciprocal ranks.

Twenty groups submitted 45 runs to the track. Unsurprisingly, for every group that submitted both 50- and 250-byte runs, the 250-byte runs was better, demonstrating that the 250-byte task is easier. The submissions from AT&T Research Labs [8] suggest that existing passage-retrieval techniques can be successful

for 250-byte runs, but are not suitable for 50-byte runs. For 50-byte runs, some explicit natural language processing (for example, entity-finding) appears necessary.

6.5 The Query track

The variability in topic performance makes it impossible to reach meaningful conclusions regarding query-dependent processing strategies unless there is a very large query set—much larger than the sets of 50 topics used in the TREC collections. The query track was designed as a means for creating a large set of different queries for an existing TREC topic set.

Participants in the track created different query versions for topics 51–100, possibly using relevance judgments from Disk 2. A query of a given type was created for each of the 50 topics, forming one query set. Four different query types were used:

Very short: two or three words extracted from the topic statement.

Sentence: an English sentence based on the topic statement and the relevant documents.

Manual feedback: an English sentence based on reading 5–10 relevant documents only (by someone who doesn't know the topic statement).

Weighted terms: a list of terms with weights (for example, as produced by an automatic feedback process).

Participants exchanged the query sets they created with all other participants in the track, and all participants ran all query sets. The document set used for the runs was the documents on Disk 1.

Since the track design included all groups running all query sets, a number of direct comparisons are possible. First, participants can see how effective their system is using their own queries. Second, they can see how effective their search component is when using other queries. Finally, participants can evaluate how effective their query construction strategies are by seeing how other groups fared with their queries.

Five groups participated in the track. There were a total of 23 different query sets produced and 9 different retrieval strategies used in the track. One of the main results of the track was confirmation of the wide variability in the effectiveness of different systems both within and across topics. One view of the data is shown in the graph in figure 9. To create the graph, the mean of the average precision scores for a system was computed over the 23 query sets for a given topic. The mean of those averages was then computed over the 9 different systems and plotted as the circle for the topic in the graph. The endpoints of the error bar for a topic represent the scores for the systems with the worst and best average scores. The mean of the average scores is a measure of the intrinsic difficulty of a topic, while the spread of an error bar represents how similarly the different systems performed.

6.6 The Spoken Document Retrieval (SDR) track

The SDR track fosters research on retrieval methodologies for spoken documents (i.e., recordings of speech). The track, which began in TREC-6, is a successor to the “confusion tracks” of earlier TREC conferences, which investigated methods for retrieving document surrogates whose true content has been confused or corrupted in some way. In the SDR track, the document surrogates are produced by speech recognition systems.

The TREC-8 track had the same general task as was used in TREC-7. A major difference between the tracks was the size of the collection used. Whereas the TREC-7 collection consisted of 87 hours of broadcast news programs, representing approximately 2900 news stories, plus 23 topics, the TREC-8 collection consisted of more than 550 hours of news broadcasts (21,500 stories) and 50 topics.

Participants worked with different versions of transcripts of the news broadcasts to judge the effects of errors in the transcripts on retrieval performance. The *reference* transcripts were based on closed captioning of the broadcasts; these transcripts were assumed to be perfect, though this assumption is less true than with the reference transcripts used in previous years. The *baseline* transcripts were produced at NIST by using NIST's installation of the BBN Rough 'N Ready BYBLOS speech recognizer. The *recognizer* transcripts were produced by the participants' own recognizer systems. The recognizer transcripts of the different

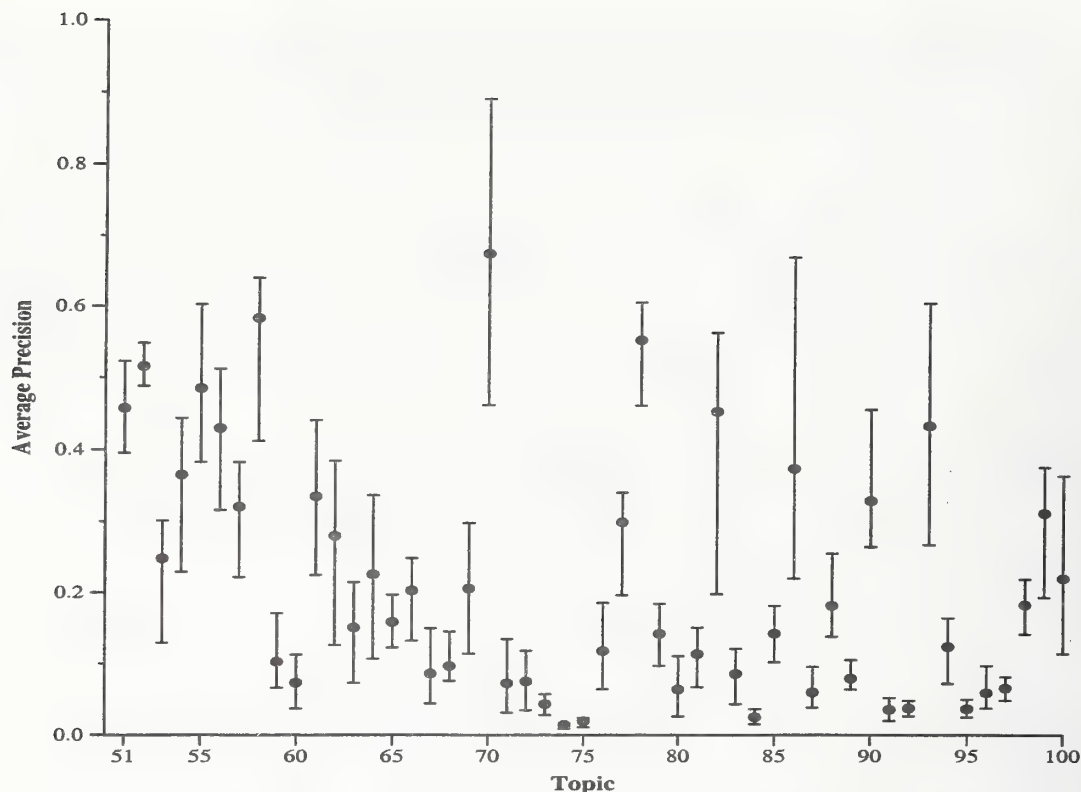


Figure 9: Average precision scores for different systems averaged over the 23 query sets per topic. The circle is the mean of the average scores computed over the 9 system variants. The error bars represent the worst and best average score for individual systems.

participants were made available to one another so that participants could perform retrieval runs against their own recognizer transcripts as well as others' recognizer transcripts (*cross-recognizer* runs). The different versions of the transcripts allowed participants to observe the effect of recognizer errors on their retrieval strategy. The different recognizer runs provide a comparison of how different recognition strategies affect retrieval.

Another difference between the TREC-7 and TREC-8 tracks was the introduction of an unknown boundary condition into the TREC-8 track. As in previous years, document boundaries were given in the reference transcripts, and these same boundaries could be used in the other versions of the transcripts as well (the known boundary case). In the unknown boundary condition, the information regarding the beginning and ending of stories was not used. Since the story boundary information is what excluded the non-news portion of the broadcasts (commercials, musical interludes, etc.) from the test collection, the unknown boundary condition entailed a much more difficult recognition task. For the unknown condition, the systems returned a ranked list of time offsets rather than a ranked list of story identifiers. During scoring, the times were mapped back to the story boundaries. Times that mapped to non-stories were assigned an invalid story identifier that was always irrelevant. Similarly, all times that mapped to a story that had already been retrieved for a topic were also assigned an invalid story identifier.

The final difference between the tracks was the provision for using a rolling language model in the participants' recognizer systems. One of the main causes of recognition errors is out-of-vocabulary words, and news stories are particularly vulnerable to this problem. To counteract this effect, participants were allowed to use a language model that (automatically) adapts to newswire texts from previous days if they desired.

Ten groups participated in the track, with six groups performing the full SDR task (recognition and retrieval) and the remaining four groups performing retrieval against the transcripts made available from NIST. In general, both speech recognition performance and retrieval performance was quite good. Retrieval

performance degraded very little for transcripts with increasing word error rates, probably due to the redundancy of key words in the spoken documents. Comparisons between the same systems run when story boundaries are known and not known show that the unknown boundary condition is more difficult, though part of the difference is that the unknown boundary runs had to process commercials and other filler material that was excluded in the known boundary case. Similar comparisons show that adaptive recognition systems can be used to more effectively recognize speech data collected over time than comparable static systems.

6.7 The Web track

Like the question answering track, the web track was a new track for TREC-8. The purpose of the track was to provide the infrastructure required to reliably evaluate new search techniques and to perform repeatable experiments in the context of the World Wide Web. The track used a frozen snapshot of the web as its document collection. This collection, known as the VLC2 collection and used in last year's Very Large Collection track [5], is over 100 gigabytes and represents some 18.5 million web pages.

The track defined two subtasks, the small web and the large web tasks, based on the amount of the web data used. The small web task used a 2 gigabyte, 250,000 document subset of the VLC2 collection, while the large web task used the entire collection.

The focus of the small web task was on answering two questions:

- Do the best methods used in the TREC ad hoc task also work best on web data? and
- Can link information in web data be used to obtain more effective search rankings than can be obtained using page content alone?

The task was exactly the same as the TREC-8 ad hoc task except that the web documents were searched instead of the documents on Disks 4 and 5. The NIST relevance assessors who judged the ad hoc pools also judged the corresponding small web pools.

Seventeen groups submitted 44 runs to the small web track. Incorporating link information did not improve retrieval performance, though that may be the result of the impoverished collection of links available in a 2 gigabyte sample of the web.

Once again we can use the test described in section 3.3.2 to gauge how complete the relevance judgments are for this collection. No manual runs contributed to the pools because the track guidelines prohibited manual runs. The total number of unique relevant documents over the 50 topics found per group ranged from a high of 89 to a minimum of 5. The mean absolute difference in mean average precision scores with and without a group's unique relevant documents computed over the 27 runs that were judged was .0021, with a maximum difference of .0073. For the 21 runs whose mean average precision score was at least .1, the mean percentage difference was 1.05 % with a maximum percentage difference of 2.85 %. These differences are quite small, and suggest that the pools were adequate to build a reliable test collection.

The large web task was also a traditional ad hoc retrieval task. In this case, however, the full VLC2 collection of documents was searched using 10,000 queries extracted from logs from the Alta Vista and Electric Monk search engines. Participants submitted the top 20 documents for all 10,000 queries to the Cooperative Research Centre for Advanced Computational Systems (ACSys). ACSys selected 50 of the 10,000 queries to judge, and judged all 20 documents for each run for those 50 queries.

The large web task is a direct descendent of the Very Large Collection track of previous years, and some of the eight groups who submitted runs to the task addressed effectiveness versus efficiency issues. Other groups used the collection to investigate distributed IR algorithms or to examine whether retrieving web documents is intrinsically different from retrieving other documents such as the newspaper articles that constitute most of the TREC collections.

7 The Future

The final session of each TREC conference is a planning session for future years. As described above, the ad hoc main task will be discontinued in TREC-9. Instead, we will focus resources on building a test collection of web documents. A 10 gigabyte sample of the VLC2 corpus will be used as the document set, and 50

topic statements will be created using real web queries taken from an Excite log as topic seeds. Pooling and relevance judging will be done as for the ad hoc task in previous years.

All of the other TREC-8 tracks will continue in TREC-9. The question answering and query tracks will perform essentially the same task as in TREC-8 so that we can gain more experience with those tasks. The task in the SDR track will also be similar, though there will be more of a focus on retrieval when document boundaries are not known. The remaining tracks will change more significantly. The cross-language track will focus on retrieving Chinese documents using English topics; research on cross-language retrieval for European languages will continue in the new CLEF initiative (see <http://www.iei.pi.cnr.it/DELOS/CLEF>). The filtering track will again have adaptive filtering, batch filtering, and routing tasks, but will use medical documents to explore how the problems change in a domain-specific environment. The task in the interactive track will change from an instance retrieval task to a question-answering task.

TREC is expected to continue beyond TREC-9. The set of tracks included in a particular year will continue to vary depending on the interests of the participants and sponsors, and the suitability of the problem to the TREC environment. New tracks will be introduced as the need arises. The call for participation for a particular TREC lists the set of tracks that it will include. The call is issued in December, and is posted on the main page of the TREC web site (<http://trec.nist.gov>) while the call is active.

Acknowledgments

The authors gratefully acknowledge the support of the TREC conferences by the Intelligent Systems Office of the Defense Advanced Research Projects Agency and by AAT. Thanks also go to the TREC program committee and the staff at NIST. The TREC tracks could not happen without the efforts of the track coordinators; our special thanks to them.

References

- [1] Chris Buckley and Janet Walz. SMART in TREC 8. In Voorhees and Harman [13].
- [2] Gordon V. Cormack, Christopher R. Palmer, and Charles L.A. Clarke. Efficient construction of large test collections. In Croft et al. [3], pages 282–289.
- [3] W. Bruce Croft, Alistair Moffat, C.J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, August 1998. ACM Press, New York.
- [4] Donna Harman. Analysis of data from the second Text REtrieval Conference (TREC-2). In *Proceedings of RIAO94*, pages 699–709, 1994.
- [5] David Hawking, Nick Craswell, and Paul Thistlewaite. Overview of the TREC-7 very large collection track. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 91–103, August 1999. NIST Special Publication 500-242. Electronic version available at <http://trec.nist.gov/pubs.html>.
- [6] Noriko Kando, Kazuko Kuriyama, Toshihiko Nozue, Koji Eguchi, Hiroyuki Kato, and Souichiro Hidaka. Overview of IR tasks at the first NTCIR workshop. In *Proceedings of the First NTCIR Workshop on Research in Japanese Text Retrieval and Term Recognition*, pages 11–44, 1999.
- [7] David D. Lewis. The TREC-4 filtering track. In D. K. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 165–180, October 1996. NIST Special Publication 500-236.
- [8] Amit Singhal, Steve Abney, Michiel Bacciani, Michael Collins, Donald Hindle, and Fernando Pereira. AT&T at TREC-8. In Voorhees and Harman [13].
- [9] K. Sparck Jones and C. van Rijsbergen. Report on the need for and provision of an “ideal” information retrieval test collection. British Library Research and Development Report 5266, Computer Laboratory, University of Cambridge, 1975.

- [10] Karen Sparck Jones. Further reflections on TREC. *Information Processing and Management*, 36(1):37–85, 2000.
- [11] Jean Tague-Sutcliffe and James Blustein. A statistical analysis of the TREC-3 data. In D. K. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3) [Proceedings of TREC-3.]*, pages 385–398, April 1995. NIST Special Publication 500-225.
- [12] Ellen M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing and Management*, 36:697–716, 2000.
- [13] E.M. Voorhees and D.K. Harman, editors. *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*. Electronic version available at <http://trec.nist.gov/pubs.html>, 2000.
- [14] Justin Zobel. How reliable are the results of large-scale information retrieval experiments? In Croft et al. [3], pages 307–314.

Cross-Language Information Retrieval (CLIR) Track Overview

Martin Braschler¹, Carol Peters², Peter Schäuble¹

¹ Eurospider Information Tech. AG, Schaffhauserstr. 18, CH-8006 Zürich, Switzerland

² Istituto Elaborazione Informazione (CNR), Via Alfieri 1, 56010 Ghezzano, Pisa, Italy

1 Introduction

A cross-language retrieval track was offered for the third time at TREC-8. The main task was the same as that of the previous year: the goal was for groups to use queries written in a single language in order to retrieve documents from a multilingual pool of documents written in many different languages. Compared to the usual definition of cross-language information retrieval, where systems work with a single language pair, retrieving documents in a language L1 using queries in language L2, this is a slightly more comprehensive task, and we feel one that more closely meets the demands of real world applications.

The document languages used were the same as for TREC-7: English, German, French and Italian. The queries were available in all of these languages. Monolingual non-English retrieval was offered to new participants who preferred to begin with an easier task. However, all the groups which did not tackle the full task opted for limited cross-language rather than monolingual runs. These experiments were evaluated by NIST and are published as unofficial ("alternate") runs. We also offered a subtask, working with documents from the field of social sciences. This collection (known as "GIRT") has some very interesting features, such as controlled vocabulary terms, title translations, and an associated multilingual thesaurus.

The track was coordinated at Eurospider Information Technology AG in Zurich. Due to its multilingual nature, the topic creation and relevance assessment tasks were distributed over four sites in different countries: NIST (English), IZ Bonn (German), IEI-CNR (Italian) and University of Zurich (French). The University of Hildesheim invested considerable effort into rendering the topics homogeneous and consistent over languages.

The participating groups experimented with a wide variety of strategies, ranging machine translation, corpus-, and dictionary-based approaches. Some results are given in Section 4. There were, however, also some striking similarities between many of the runs, such as the choice of English as topic language the majority, and the use of Systran by a lot of groups. Some implications of these findings are discussed in Section 5.

The main goal of the TREC CLIR activities has been the creation of a multilingual test collection that is re-usable for a wide range of evaluation experiments. This means that the quality of the relevance assessments is very important. The Twenty-One group conducted an interesting analysis with respect to the completeness of the assessments and the impact of this on the pool. We address some of their findings in Section 5.

The paper concludes with an indication of our plans for the future of the cross-language track, which will bring substantial changes to the format and coordination of the activities.

2 Overview of CLIR

There are three main ways in which cross-language information retrieval approaches attempt to "cross the language barrier" – through query translation, or document translation, or both. (Oard, 1997). CLIR research started out with experiments using controlled vocabularies and associated dictionaries and thesauri, but nowadays free text approaches are most common. These approaches also dominate experiments in past and present CLIR tracks. Free text methods can be further classified according to the resources used to cross the language boundary: machine translation, machine-readable dictionaries, or corpus-based resources.

Machine translation (MT) seems an obvious choice for cross-language information retrieval systems. It also played a large role in the TREC-8 experiments of a number of groups. However, CLIR is a difficult problem to solve on the basis of MT alone: queries that users typically enter into a retrieval system are rarely complete sentences and provide little context for sense disambiguation.

Corpus-based approaches are also popular. Groups experimenting with such approaches during this or former CLIR tracks include Eurospider, IBM and the University of Montreal.

Lastly, a significant number of cross-language retrieval approaches make use of existing linguistic resources, mainly machine-readable bilingual dictionaries. Various ideas have been proposed to address some of the problems associated with dictionary-based translations, such as ambiguities and vocabulary coverage. One of the groups that have investigated the use of such dictionaries is the Twenty-One consortium.

3 CLIR-Track Task Description

Similarly to last year, CLIR track participants were asked to retrieve documents from a multilingual pool containing documents in four different languages. They were free to choose the topic language, and then had to find relevant documents in the pool regardless of the languages in which the texts were formulated. Most groups approached this task by performing separate bilingual retrieval runs, and then combining the results. The merging of their retrieval results was therefore an additional problem for these groups.

Documents for TREC-8 were in English, German, French and Italian. There were 28 topics, each one provided in all four languages. In order to attract newcomers, monolingual non-English runs were accepted; however, participants preferred to do bilingual cross-language runs when they could not do the full task.

The TREC-8 task description also included a vertical domain subtask, working with a second data collection, containing documents from a structured database in the field of social science (the "GIRT" collection). This collection comes with English titles for most documents, and a matching bilingual thesaurus. The University of Berkeley conducted some very extensive experiments with this collection.

The document collection for the main task contained mainly news-wire articles. The English texts were taken from three years (1988 to 1990) of Associated Press news stories. For German, French and Italian, news stories were taken from SDA, the "Schweizerische Depeschagentur" (Swiss News Agency), covering the same time period. While these texts were produced by the same agency, this does not mean that they contain actual translations. However, there is a sizeable topic overlap between the texts in the three languages, enabling experiments with alignment on these collections (for example experiments by Eurospider and IBM). For German, texts from the Swiss newspaper "Neue Zürcher Zeitung" (NZZ) for 1994 were also added. Table 1 gives more details on the document collections.

Document collections			
Language	Source	No. Documents	Size
English	AP news, 1988-90	242,918	750 MB
German	SDA news, 1988-90	185,099	330 MB
	NZZ articles, 1994	66,741	200 MB
French	SDA news, 1988-90	141,656	250 MB
Italian	SDA news, 1989-90	62,359	90 MB

Table 1: figures for the document collections.

For TREC-6, the CLIR track topics were developed centrally at NIST (Schäuble and Sheridan, 1998). However, problems during the topic creation and relevance assessment process and reactions from participants showed that this was not an optimal solution. A good translation has to take regional and cultural differences into account, and this is very hard to achieve if there is just one topic creation site. Consequently, in TREC-7, a distributed topic creation and relevance assessment setup was introduced (Braschler et al., 1999). This made it much easier to use native speakers in the translation stage which helped to improve overall quality. However, spreading this process over several sites means increased coordination overheads. The danger of producing inconsistent translations was addressed by active communication between the sites through e-mail and meetings. We retained this distributed setup for TREC-8. In addition, we received valuable help from University of Hildesheim in ensuring the consistency and quality of the topics.

The topic creation and results assessment sites for TREC-8 were:

- English: NIST, Gaithersburg, MD, USA (Ellen Voorhees)
- French: University of Zurich, Switzerland (Michael Hess)
- German: IZ Sozialwissenschaften, Germany (Jürgen Krause, Michael Kluck)
- Italian: IEI-CNR, Pisa, Italy (Carol Peters)

At each site, an initial 10 topics were formulated. At a topic selection meeting, the seven topics from each site that were felt to be best suited for the multilingual retrieval setting were then selected. Each site then translated the 21 topics formulated by the others into the local language. This ultimately led to a pool of 28 topics, each available in all four languages. It was decided that roughly one third of the topics should address national/regional, European and international issues, respectively. To ensure that topics were not too broad or too narrow and were easily interpretable against all document collections, monolingual test searches were conducted.

Participants were free to experiment with different topic fields (using either the title, description or narrative – or all three), and with both automatic and manual runs, similar to the definitions of the TREC adhoc task.

4 Results

A total of twelve groups from six different countries submitted results for the TREC-8 CLIR track (see Table 2). Eight participants tackled the full task (up from last year's five), submitting 27 runs (up from 17). The remainder of the participants either submitted runs using a subset of languages, or concentrated on the GIRT subtask only. English was the dominant topic language,

even more so than last year. This development was not anticipated in such a pronounced form. Still, each language was used by at least one group as the topic language.

Participant	Country
Claritech	USA
Eurospider Information Technology AG	Switzerland
IBM	USA
IRIT/SIG	France
Johns Hopkins University APL	USA
MNIS-Textwise Labs	USA
New Mexico State University	USA
Sharp Laboratories of Europe Ltd	UK
Twenty-One	Netherlands
University of California, Berkeley	USA
University of Maryland	USA
University of Montreal	Canada

Table 2: Distribution of participants.

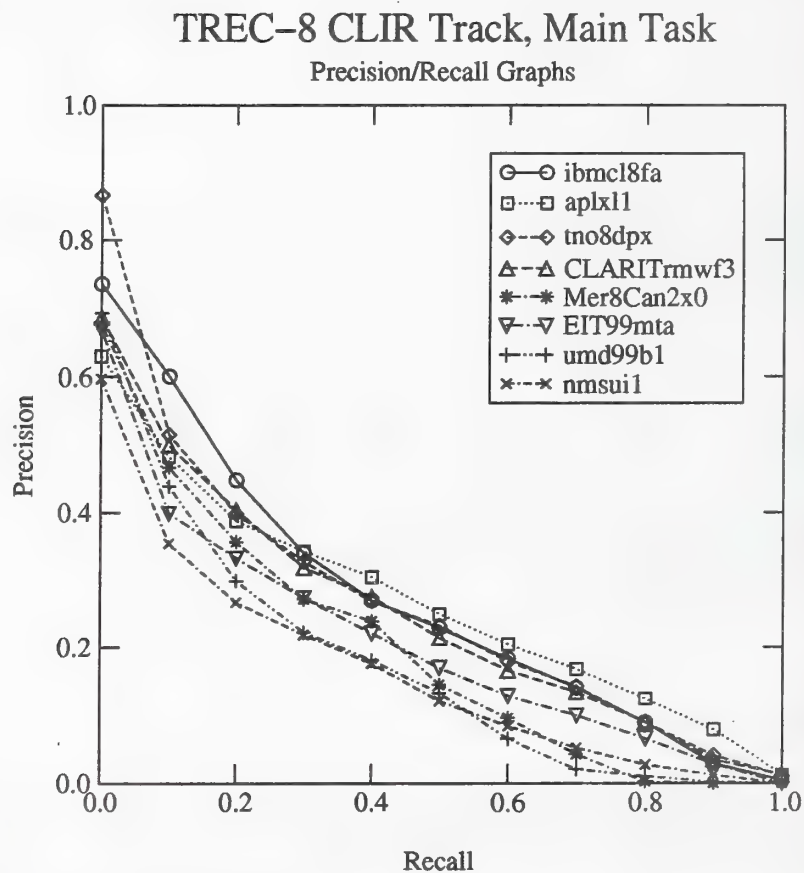


Figure 1: Runs for the main task

The relevance assessments used for the evaluation of these runs were performed by the same four sites listed above.

While the average precision numbers improved in TREC-7 with respect to TREC-6, they fell slightly in TREC-8; this is perhaps due to having a smaller average number of relevant documents per topic.

Figure 1 shows a comparison of runs for the main task. The graph shows the best automatic runs against the full document pool for each of the eight groups. Because of the diversity of the experiments conducted, the figures are best compared on the basis of the specific features of the individual runs. These can be found in the track papers. For example, New Mexico State runs use manually translated queries, which are the result of a monolingual user interactively picking good terms. This is clearly an experiment that is very different from the runs of some other groups that are essentially doing "ad-hoc" style cross-language retrieval, using no manual intervention whatever.

Approaches employed in TREC-8 by individual groups include:

- experiments on pseudo relevance feedback by Claritech (Qu et al., 2000)
- similarity thesaurus based translation by Eurospider (Braschler et al., 2000)
- statistical machine translation by IBM (Franz et al., 2000)
- combinations of n-grams and words by JHU (Mayfield et al., 2000)
- use of conceptual interlingua by Textwise (Ruiz et al., 2000)
- query translation using bilingual dictionaries by Twenty-One (Kraaij et al., 2000)
- evaluation of the Pirkola measure by University of Maryland (Oard et al., 2000)
- transaction models derived from parallel text by University of Montreal (Nie, 2000)
- use of an online machine translation system by Mercure/IRIT (Boughanem et al., 2000)

This diversity of approaches is one of the characteristics that makes the CLIR track extremely interesting and shows that there is still a lot of room for further studies and development.

Merging remained an important issue for most participants. University of Maryland tried to circumvent the problem by using an unified index in some of their runs, but the other groups working on the main task all had to rely on merging of some sort to combine their individual, bilingual cross-language runs. Some of the approaches this year include: merging based on probabilities - calculated using log(Rank) by various groups including IBM, merging using linear regression on document alignments by Eurospider, linear combinations of scores by JHU, and of course, straight, score-based merging.

Two groups submitted runs for the GIRT subtask. Berkeley even participated exclusively in the subtask only, and did some very comprehensive experiments using both the English titles of the documents and the English/German thesaurus supplied with the collection (Gey and Jiang, 2000). These runs show some of the interesting properties of GIRT, and we hope that this subtask will have more participants in the future.

It is also possible to do ad-hoc style runs on GIRT, ignoring controlled vocabulary, English titles and the thesaurus. This approach was taken by Eurospider.

5 Observations and Trends

It is interesting to note certain similarities between the submissions of a number of participants this year. Two main points stand out with respect to the main task: first, 21 out of

27 submitted runs used English as the topic language, and second, that at least half of all groups used the Systran machine translation system in some form for parts of their experiments.

Although it is not surprising that English is a popular choice as topic language, we did not expect this language to be so dominant. While English was also the most popular choice for TREC-7, the percentage of runs that used non-English topics was substantially higher (7 out of 17). We had hoped that with the CLIR track in its third year, more groups would start to experiment with non-English query languages. That this has not been the case could be due to several factors. The fact that three quarters of the participants are located in English speaking countries certainly plays an important role. If we can encourage more European groups to participate in this activity, the ratio should become more balanced.

However, we believe it is also a result of a lack of resources available to some of the groups. The coordinators have always been aware that the main task of handling four languages may appear daunting to newcomers. In the past, we attempted to lessen the "shock" by allowing either cross-language runs on subsets of languages, or monolingual non-English runs. The intention was to allow groups that did not have access to resources for all languages, or were lacking experience in handling some of the languages, to start slowly and then expand their participation in the future.

While it is encouraging to see that most groups did try to tackle the main task, the fact that the majority of them chose English as their topic language may indicate that they are still constrained in the kind of resources available to them. They may have found dictionaries for English and the other languages, but not for e.g. German to Italian. The resource problem therefore seems to remain as a stumbling block. In the future, we hope to invest some efforts into building a repository for such resources that will allow participants to share whatever free components they have available. Together with the continued offer to start with easy tasks, this should also contribute to encouraging new groups to participate in cross-language system evaluation activities.

Similarly, we feel that part of the reason for the choice of Systran by so many groups also lies in a lack of resources: using Systran allowed the groups to do at least something with certain language pairs that they would otherwise not have been able to include in their experiments. That Systran offers mainly combinations of English with other languages probably also contributed to the domination of English as topic language.

Another area that merits attention this year is that of the relevance assessments. The Twenty-One group made an interesting analysis of the TREC-7 pool of relevance judgments. The quality of the pool and the judgments was also a topic of discussion on the mailing list leading up to the TREC-8 conference. The literature reports a considerable number of interesting experiments aimed at testing the quality and the properties of relevance assessments. The work by Voorhees (Voorhees, 1998) is particularly notable. Working with the relevance assessments of the TREC-4 and TREC-6 ad-hoc task, Voorhees found that the relative effectiveness of different retrieval strategies remains stable despite marked differences in the relevance judgments used to measure retrieval. This means that while the actual values of the effectiveness measure (i.e. average precision) are affected by differences in relevance judgements, the relative retrieval performance remains almost always constant. While the analysis by the Twenty-One group was concerned with a slightly different question, namely if the size of the pool is sufficient, we felt it would be interesting to spot-check the hypothesis that the ordering remain mostly stable even when the values of the relevance judgments are altered. In fact, we found that, on the basis of the numbers given by Twenty-One in their paper, the ranking of the systems would probably have remained nearly identical, even if individual runs were not judged. Since the runs that were analyzed by Twenty-One are a mix of multilingual and bilingual experiments, and since it was not possible to re-run all the experiments in time for

this paper, unfortunately, we cannot give exact figures. However, the only two runs that seem to have any real potential for changing ranks are the RaliDicAPf2e and ceat7f2 runs. As can be seen from the numbers given in the Twenty-One paper, these are the two runs that provide the most unique relevant documents. They are also very close to some other runs in their absolute values. These two factors combine to increase the probability of a change in ranking. Note also that for the three groups that had multiple runs judged (Berkeley, Eurospider and Twenty-One), the ordering of the runs does not change in any case. This is consistent with the findings of Voorhees for the TREC-style relevance judgments analyzed in her paper, where she states that comparing algorithmic variants of the same system is very reliable.

Constantly questioning the relevance assessments and analyzing their quality remains very important when the goal is to create a reliable test suite for cross-language system evaluation. Most research on the topic is encouraging, and the considerations outlined above that indicate a stable ranking seems to imply that such findings are also valid in the case of the cross-language pool. We have to remain vigilant with respect to the quality of that pool since, as the Twenty-One group points out, it is still rather small. We are however confident that participants receive valuable results from their evaluation through the CLIR track. It is certainly true that non-participants might have more difficulties in interpreting their results based on the small size of the CLIR pool, as Twenty-One points out. We hope, however, that this will encourage these people to participate in the future, thus increasing the size of the pool. This is the best way to improve the pool.

6 Move to Europe and CLEF

From 2000 on, it has been decided to coordinate cross-language system evaluation for what are traditionally considered as European languages in Europe rather than in the U.S, although still in collaboration with NIST and TREC. The European side is sponsored by the DELOS Network of Excellence for Digital Libraries and funded by the European Commission.

There are several reasons that have led to this decision. Perhaps the main one is that, as already mentioned, much of the work was already being done in Europe. However, moving the coordination to Europe not only makes logistic sense but also leaves NIST freer to concentrate on cross-language evaluation on other language groups. In fact, in 2000, TREC will be offering a cross-language track using English and Mandarin documents and English topics. Depending on data availability, the track may also involve Tamil and Malay documents.

More importantly, this move and the launching of an independent activity – known as CLEF (for Cross-Language Evaluation Forum) – allows us to focus on a wider range of issues. As has been stated, the main task offered in TREC-7 and 8 – the multilingual retrieval task – was a hard task and possibly discouraged some potential participants who did not have the resources (or the confidence) to tackle cross-language retrieval with all four languages. Thus, we have decided to provide a greater variety of tasks in CLEF 2000. The aim is both to encourage the participation of groups who are only now beginning to tackle the issues involved in cross-language retrieval, and also to extend the possibility of participation to groups developing systems for other European languages.

There will thus be three main evaluation tasks in CLEF 2000: multilingual information retrieval, bilingual information retrieval, and monolingual (non-English) retrieval, plus again the GIRT sub-task for cross-language retrieval in a special domain. Interested groups can participate in any one or in all four tracks.

Similarly to TREC-8, the main task of CLEF 2000 requires searching a multilingual document collection for relevant documents, and listing the results in a merged, ranked list. Although the official languages are again English, French, German and Italian, it is also possible to submit runs in which the document collection is queried in other languages. In this

case, participants will be responsible for the translation of the query into their selected language. The results for such runs will be given separately. A pair-wise cross-language task is provided in which the query language can be French, German or Italian and the target document collection is English. Many IR groups are now beginning to work on retrieval over pairs of languages and this will give them a chance to participate officially in the CLEF activity. Unofficial bilingual runs in which the query to the English document collection can be in any other European language can also be submitted and will be evaluated.

Multilingual information retrieval implies a good understanding of the issues involved in monolingual retrieval. It is often asserted that procedures for monolingual information retrieval are (almost) completely language independent. This is not however true; different languages present different problems. Methods that may be highly efficient for certain language typologies may not be so effective for others. Issues that have to be catered for include word order, morphology, diacritic characters, language variants. So far, most IR system evaluation has focussed on English. CLEF will provide the opportunity for monolingual system testing and tuning and build up test suites in other European languages (beginning with French, German and Italian in CLEF 2000).

The CLEF multilingual document pool for 2000 consists of comparable corpus consisting national newspapers for all four languages from the same time period; a change from the news agency stories of previous years. Topics will be developed much as before; however, the use of Italian French and German national papers rather than Swiss sources will perhaps extend the multicultural aspect. It is hoped to be able to offer additional languages in future years. The number of topics will be increased with the aim of building up the size of the pool as quickly as possible.

The results of CLEF 2000 will be presented at a two-day workshop to be held in September in Lisbon, Portugal, immediately after the fourth European Conference on Digital Libraries (ECDL 2000). The first day will be open to all interested participants and focussed on research related issues in Multilingual Information Access. The second day will report and discuss the results of the CLEF activity and will be restricted to active CLEF participants.

More information on CLEF can be found at <http://www.iei.pi.cnr.it/DELOS/CLEF/>.

Acknowledgements

Our thanks go to the data providers, Neue Zürcher Zeitung (NZZ), Schweizerische Depeschagentur (SDA) and the Associated Press (AP). We would also like to express our gratitude to the topic creation and relevance assessment sites for their great work.

References

- Boughanem, M., Julien, C., Mothe, J., and Soule-Dupuy C. (2000). Mercure at trec8: Adhoc, Web, CLIR and Filtering tasks. In *Proceedings of the Eighth Text Retrieval Conference (TREC8)*.
- Braschler, M., Kan, M.-Y., Schäuble, P., and Klavans, J. (2000). The Eurospider Retrieval System and the TREC-8 Cross-Language Track. In *Proceedings of the Eighth Text Retrieval Conference (TREC8)*.

- Braschler, M., Krause, J., Peters, C., and Schäuble, P. (1999). Cross-Language Information Retrieval (CLIR) Track Overview. In *Proceedings of the Seventh Text Retrieval Conference (TREC7)*.
- Franz, M., McCarley, J. S., and Ward, R. T. (2000). Ad hoc, Cross-language and Spoken Document Information Retrieval at IBM. In *Proceedings of the Eighth Text Retrieval Conference (TREC8)*.
- Gey, F. C. and Jiang, H. (2000). English-German Cross-Language Retrieval for the GIRT Collection - Exploiting a Multilingual Thesaurus. In *Proceedings of the Eighth Text Retrieval Conference (TREC8)*.
- Kraaij, W., Pohlmann, R., and Hiemstra, D. (2000). Twenty-One at TREC-8: using Language Technology for Information Retrieval. In *Proceedings of the Eighth Text Retrieval Conference (TREC8)*.
- Mayfield, J., McNamee, P., and Piatko, C. (2000). The JHU/APL HAIRCUT System at TREC-8. In *Proceedings of the Eighth Text Retrieval Conference (TREC8)*.
- Nie, J.-Y. (2000). CLIR using a Probabilistic Translation Model based on Web Documents.
- Oard, D. W. (1997). Cross-Language Text Retrieval Research in the USA. Presented at 3rd *ERCIM DELOS Workshop, Zurich, Switzerland*.
Available from <http://www.clis.umd.edu/dlrg/filter/papers/delos.ps>
- Oard, D. W., Wang, J., Lin, D., and Soboroff, I. (2000). TREC-8 Experiments at Maryland: CLIR, QA and Routing. In *Proceedings of the Eighth Text Retrieval Conference (TREC8)*.
- Qu, Y., Jin, H., Eilerman, A. N., Stoica, E., and Evans D. A. (2000). CLARIT TREC-8 CLIR Experiments. In *Proceedings of the Eighth Text Retrieval Conference (TREC8)*.
- Ruiz, M., Diekema, A., and Sheridan, P. (2000). CINDOR Conceptual Interlingua Document Retrieval: TREC-8 Evaluation. In *Proceedings of the Eighth Text Retrieval Conference (TREC8)*.
- Schäuble, P. and Sheridan, P. (1998). Cross-Language Information Retrieval (CLIR) Track Overview. In *Proceedings of the Sixth Text Retrieval Conference (TREC6)*.
- Voorhees, E. M. (1998). Variations in Relevance Judgments and the Measurement of Retrieval Effectiveness. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

The TREC-8 Filtering Track Final Report

David A. Hull
Xerox Research Centre Europe
Meylan, France
hull@xrce.xerox.com

Stephen Robertson
Microsoft Research
Cambridge, UK
ser@microsoft.com

Abstract

The TREC-8 filtering track measures the ability of systems to build persistent user profiles which successfully separate relevant and non-relevant documents. It consists of three major subtasks: adaptive filtering, batch filtering, and routing. In adaptive filtering, the system begins with only a topic statement and must learn a better profile from on-line feedback. Batch filtering and routing are more traditional machine learning tasks where the system begins with a large sample of evaluated training documents. This report describes the track, presents the evaluation results in graphical format, and provides a general commentary on lessons learned from this year's track.

1 Introduction

A text filtering system sifts through a stream of arriving information to find documents relevant to a set of user profiles. Unlike the traditional search query, user profiles are persistent, and tend to reflect a long term information need. With user feedback, the system can learn a better profile, and improve its performance over time. The TREC filtering track tries to simulate on-line time-critical text filtering applications, where the value of a document decays rapidly with time. This means that potentially relevant documents must be presented immediately to the user. There is no time to accumulate and rank a set of documents according to their relevance. Evaluation is based only on the quality of the retrieved set, which is scored using a utility measure. The utility measure assigns a positive score for each relevant document retrieved and a negative score to each retrieved document that is not relevant.

Filtering differs from search in that documents arrive sequentially over time. The TREC filtering track consists of three subtasks: adaptive filtering, batch filtering, and routing. In adaptive filtering, the system starts with only a user profile and must begin filtering documents without any other prior information. Each retrieved document is immediately judged for relevance, and this information can be used by the system to adaptively update the filtering profile. In batch filtering and routing, the system starts with a large set of evaluated training documents which can be used to help construct the search profile. For batch filtering, the system must decide to accept or reject each document, while routing systems can return a ranked list of documents. The core tasks have remained the same in TREC-7 and TREC-8.

Traditional adhoc retrieval and routing simulate a non-interactive process where users look at documents once at the end of system processing. This allows for ranking or clustering of the retrieved set. The filtering model is based on the assumption that users examine documents periodically over time. The actual frequency of user interaction is unknown and task-dependent. Rather than create a complex simulation which includes partial batching and ranking of the document set,

we make the simplifying assumption that users want to be notified about interesting documents as soon as they arrive. Therefore, a decision must be made about each document without reference to future documents, and the retrieved set is ordered by time, not estimated likelihood of relevance. The history and development of the TREC Filtering Track can be traced by reading the yearly final reports:

- TREC-7 http://trec.nist.gov/pubs/trec7/t7_proceedings.html (#3 - 2 files) [3]
- TREC-6 http://trec.nist.gov/pubs/trec6/t6_proceedings.html (#4 and #5) [2]
- TREC-5 http://trec.nist.gov/pubs/trec5/t5_proceedings.html (#5) [5]
- TREC-4 http://trec.nist.gov/pubs/trec4/t4_proceedings.html (#11) [4]

Information on the participating groups and their filtering systems can be found in the individual site reports, also available from the TREC web site.

2 TREC-8 Task Description

The basic filtering tasks have not changed from TREC-7 to TREC-8, so readers familiar with the TREC-7 task may wish to skip this section. In this section, we review the corpus, the three sub-tasks, the submission requirements, and the evaluation measures. For more background and motivation, please consult the TREC-7 track report [3]. The TREC-8 filtering experiments used the Financial Times document collection (TREC disk 4), which consists of slightly more than three years of newspaper articles covering part of 1991 and most of 1992-1994. The 210,000 documents were ordered roughly as a function of time, and all systems were required to process the collection (or a subset) in the same order. The documents average 412 words in length and cover a wide variety of subject matter. All tasks used topics 351-400, which were constructed for the TREC-7 adhoc experiments. The topics contain Title, Description, and Narrative fields and have an average length of 58 words.

The adaptive filtering task is designed to model the text filtering process from the moment of profile construction. No training documents are provided. However, once a document is retrieved, the relevance assessment (when one exists) is immediately made available to the system. Unfortunately, it is not feasible in practice to have interactive human assessment by NIST.¹ Instead, assessment is simulated by releasing the pre-existing relevance judgement for that document. Judgements for unretrieved documents are never revealed to the system. Once the system makes a decision about whether or not to retrieve a document, that decision is final. No back-tracking or temporary caching of documents is allowed. While not always realistic, this condition reduces the complexity of the task and makes it easier to compare performance between different systems.

Systems are allowed to use the rest of the TREC document collection (excluding the Financial Times) to generate collection frequency statistics (such as IDF) or auxiliary data structures (such as automatically-generated thesauri). While access to relevance judgements for topics 351-400 was restricted to the final run, systems could be trained using other topics on the rest of the TREC collection. As documents were processed, the text could be used to update term frequency statistics and auxiliary document structures even if the document was not matched to any profile. Groups had the option to treat unevaluated documents as not relevant. Evaluation is based on utility, as described in the next section.

¹Individual participants have the option to assess documents manually, although all such runs are evaluated in a different category. No groups submitted manual runs for TREC-8.

In batch filtering, documents and relevance judgements from the 1991-1992 Financial Times are available in advance as a training set. The 1993-1994 Financial Times documents form the test set. As in adaptive filtering, systems may use the relevance judgement from any retrieved document to update the filtering profile. Evaluation is based on utility. Routing is similar to batch filtering, with the following two differences. Systems are allowed to use relevance judgements for topics 351-400 from other parts of the TREC document collection as part of the training set. For routing, systems return a ranked list of the top 1000 retrieved documents and are evaluated according to average uninterpolated precision, as in adhoc search. Batch filtering and routing are included to open participation to as many different groups as possible and to improve the quality of the document pool used for evaluation.

2.1 Evaluation

For the TREC experiments, filtering systems are expected to make a binary decision to accept or reject a document for each profile. Therefore, the retrieved set consists of an unranked list of documents. The primary evaluation measure is utility. Utility assigns a value or cost to each document, based on whether it is retrieved or not retrieved and whether it is relevant or not relevant. For linear utility, the score is a linear combination of the elements in the contingency table shown below:

	Relevant	Not Relevant
Retrieved	R+ / A	N+ / B
Not Retrieved	R- / C	N- / D

$$\text{Linear Utility} = A \cdot R+ + B \cdot N+ + C \cdot R- + D \cdot N-$$

The variables R+/R-/N+/N- refer to the number of documents in each category. The utility parameters (A,B,C,D) determine the relative value of each possible category. A positive utility parameter can be thought of as the value of each document in that category, while a negative utility parameter is the cost of classifying a document in that category. Therefore, the larger the utility score, the better the filtering system is performing for a given query profile. For TREC-8, we use two different linear utility functions:

$$\begin{aligned} \text{LF1} &= 3 \cdot R+ - 2 \cdot N+ && \text{--> retrieve if } P(\text{rel}) > .4 \\ \text{LF2} &= 3 \cdot R+ - N+ && \text{--> retrieve if } P(\text{rel}) > .25 \end{aligned}$$

Filtering according to a utility function is equivalent to filtering by estimated probability of relevance. Therefore, the utility functions above are listed with the appropriate probability thresholds.

In addition, we tested the following non-linear utility functions for the first time at TREC-8:

$$\text{NF1} = 6 \cdot (R+)^{0.5} - N+$$

$$\text{NF2} = 6 \cdot (R+)^{0.8} - N+$$

The idea behind the non-linear utility functions (originally due to I.J. Good [1]) is that the value of a relevant document depends on how many relevant documents have been seen before. In this formulation, the more relevant documents retrieved, the lower the value of each additional relevant document. From a practical perspective, our hope is that the non-linear functions will do a better job of equalizing the size of the retrieved sets for topics with large differences in the number of relevant documents.

When evaluation is based on utility, it is difficult to compare performance across topics. Simple averaging of the utility measure gives each retrieved document equal weight, which means that the average scores will be dominated by the topics with large retrieved sets (as in micro-averaging). Therefore, we scale the utility scores prior to averaging. The most obvious scaling strategy is to divide by the maximum possible utility score for each topic. However, this approach is seriously flawed for negative utility scores. A system which returns one hundred non-relevant documents will receive a score of -100 for a topic with one relevant document and -1 for a topic with one-hundred relevant documents. Since the maximum possible positive score on a topic is 1, topics with negative utilities will dominate the average. Therefore, a more complex utility scaling function is required.

The scaling function used for TREC-8 is:

$$u_s^*(S, T) = \frac{\max(u(S, T), U(s)) - U(s)}{MaxU(T) - U(s)}$$

where $u(S, T)$ and $u_s^*(S, T)$ are the original and scaled utility of system S for topic T , $U(s)$ is the utility of retrieving s non-relevant documents, and $MaxU(T)$ is the maximum possible utility score for topic T . This scaling function assigns a lower bound to the utility function which can be set with the parameter s . There is a reasonable justification for this approach. Assume that a filtering profile is performing really poorly. At some point, the user will get fed up with reading non-relevant documents and delete the profile entirely. The parameter s sets the number of non-relevant documents at which the user's tolerance is exhausted. All utility scores less than $U(s)$ are set to $U(s)$. Therefore, utility scores can range between $U(s)$ and $MaxU(T)$ and the scores are renormalized to range between 0 and 1 and then averaged.

The parameter s is set once for all topics. The scaled score can be interpreted relative to the perfect system (utility of 1.0) and the worst possible system (0.0). Unfortunately, the average scaled utility score is highly dependent on the definition of the worst possible system, as determined by the parameter s , and so is only meaningful in relation to a particular lower bound. Since the decision on where to put the lower bound is fairly arbitrary, we will plot the average scaled utility scores over a range of values for s (25-800). A low value for s differentiates more between systems that do well on topics with few relevant documents. On the other hand, it reduces the penalty for systems which do very poorly on these or other topics. One could define a two-parameter scaling model which sets the zero point for utility scaling and the lower bound for acceptable performance to different values, thus allowing for negative scaled utility. This would make it possible to distinguish to the two types of behavior described above. For simplicity, we choose to use the same value for the zero point and the lower bound in these experiments.

2.1.1 Evaluation measures – further discussion

During the period between TREC-7 and TREC-8, there was some continuing discussion of the problem of devising suitable evaluation measures for filtering tasks. This discussion followed several different lines, and by no means all the contentious issues were resolved. This section attempts some kind of summary of parts of the discussion. The reasons for the introduction of the utility scaling function and of the non-linear utility functions were discussed above. Both ideas were adopted in the plan for the TREC-8 filtering track, and were included in the guidelines. However, both ideas came into some question in the subsequent discussion on the track mailing list.

One characteristic of utility in general that attracted comment was its relationship to recall and precision. With any utility function, it is possible to imagine two systems X and Y which have the following characteristics:

$$\text{Precision}(X) > \text{Precision}(Y)$$

$$\begin{aligned}\text{Recall}(X) &> \text{Recall}(Y) \\ \text{but } U(X) &< U(Y)\end{aligned}$$

With the linear functions, this arises only when both utilities are negative; this can be demonstrated as follows. Suppose the system retrieves R^+ relevant and N^+ non-relevant documents (as above), and (again as above) the utility function is:

$$U = AR^+ + BN^+ = R^+(A + B\frac{N^+}{R^+})$$

(B will of course be negative). Then we assume for precision that $\frac{R_X^+}{(R_X^+ + N_X^+)} > \frac{R_Y^+}{(R_Y^+ + N_Y^+)}$ which is equivalent to $\frac{N_X^+}{R_X^+} < \frac{N_Y^+}{R_Y^+}$. For recall we have $R_X^+ > R_Y^+$. If $U(Y)$ is positive (which implies that $A + B\frac{N_Y^+}{R_Y^+}$ is positive), then

$$\begin{aligned}U(X) &= R_X^+(A + B\frac{N_X^+}{R_X^+}) \\ &> R_Y^+(A + B\frac{N_X^+}{R_X^+}) \\ &> R_Y^+(A + B\frac{N_Y^+}{R_Y^+}) \text{ since } B < 0 \\ &= U(Y)\end{aligned}$$

However, if $U(X)$ is negative, we can infer that $U(Y)$ is also negative, but that the above inequalities are reversed, and $U(X) < U(Y)$. Non-linear utility functions exhibit similar behaviour under different conditions, and in particular may do so in the positive range. The behaviour was in fact in evidence in some of the TREC-8 results (see section 3.3).

The significance of this fact was the subject of some debate in the group. It is clear that this reflects a genuine characteristic of user preference, provided that user preference is well described by the utility function. However, it also implies that a system might be improved by introducing a random element to it, which suggests that what we are measuring is less a characteristic of a system and more of a particular user preference.

Also, one justification for the scaling function is that the zero of the utility scale is essentially arbitrary: given any decision rule based on probability of relevance, such as those indicated in section 2.1 above, it could be derived from any number of different utility functions with different zero points. This suggests that an arbitrary redefinition of the zero point (which the scaling function achieves by setting the utility of the worst possible system to zero) will not interfere too much with the essential properties of utility. However, the present argument casts doubt on this conclusion, because even if two different utility functions with two different zeros generate the same decision rule, they will have different relationships to recall and precision. Scaling by the method chosen is *not* equivalent to choosing a utility function with a different zero, and produces different behaviour.

This discussion took place shortly before TREC-8, and therefore did not affect the track guidelines or way the results were evaluated. However, it is clear that the issues raised will have to be revisited for TREC-9.

2.2 Submission Requirements

Each participating group could submit up to six adaptive filtering runs and up to two runs each for batch filtering and routing. Each adaptive or batch filtering run was evaluated according to a

pre-specified utility function. There were no required runs this year, although we asked each group that participated in adaptive or batch filtering to submit one run optimized for the LF1 measure. Runs were classified into one of three categories:

- (A) Automatic - Any run which uses fully automatic methods for profile construction and updating. This can include automatic learning from test documents as they are filtered.
- (B) Manual - Any run which uses manual techniques for profile construction, up to and including making additional relevance judgments on training documents. No manual intervention based on information from the test documents is allowed, although automatic learning is still permitted.
- (C) Manual Feedback - Any run which uses manual techniques for updating profiles based on previously viewed test documents. The run may or may not also use manual techniques for profile construction.

There are no training documents for adaptive filtering, so manual intervention (B) is limited to profile construction for this task. In practice, none of the groups submitted manual runs. Groups were also asked to indicate whether they used other parts of the TREC collection to build term collection statistics or other resources.

Topics 351-400 were evaluated against the Financial Times collection for the TREC-7 adhoc experiments, and these relevance judgements served as the basis for the filtering tasks. However, we wanted to reduce the risk that systems which retrieved many unevaluated documents might be unfairly penalized, so we asked NIST to perform a second round of evaluation. We are grateful to NIST for following through on this request, given the heavy assessment load for TREC-8. In the end, NIST managed to evaluate the top 57 documents from every submitted run, although documents with existing relevance judgement were not re-assessed. A significant number of new relevant documents were found in the second round of assessment, indicating that it was a worthwhile endeavor².

3 TREC-8 results

Fourteen groups participated in the TREC-8 filtering track and submitted a total of 55 runs. This represents roughly a 20% increase over the TREC-7 participation level.

	# groups	# runs
Total	14	55
-----	--	--
adaptive	8	33
batch	6	11
routing	6	11

Here is a list of the participating groups, including [abbreviations] and (run identifiers). Participants will generally be referred to by their abbreviations in this paper. The run identifiers can be used to recognize which runs belong to which groups in the plotted results.

- City Univ. London / Microsoft [City] (plt8f)
- CLARITECH Corporation [CLARITECH] (CL99)

²As reported in the overview presentation of the TREC-8 Conference by Ellen Voorhees.

Organization	Runs	1st yr?	Adaptive	Batch	Routing
City	plt8f	-	-	X	X
CLARITECH	CL99	-	X	X	-
DSO	dso99r	X	-	-	X
ICDC	S2N2	X	-	-	X
IRIT	Mer8	-	-	X	X
KDD	kdd8f	X	X	-	-
Microsoft	ok8f	-	X	-	-
CUNY	pir9	-	X	X	X
Rutgers-K	Ant	-	X	X	-
Seoul	Scai8	X	-	X	-
TNO	uttno8	-	X	-	-
UIowa	IOWAF	-	X	-	-
UMass	INQ	-	X	-	-
UMaryland	umr	X	-	-	X

Table 1: Summary of task participation (X = participant).

- DSO National Laboratories, Singapore [DSO] (dso99r)
- Informatique-CDC - Groupe Caisse des Dépôts / ESPCI [ICDC] (S2N2)
- IRIT / University of Toulouse (IRIT) (Mer8)
- KDD [KDD] (kdd8f)
- Microsoft Research - Cambridge [Microsoft] (ok8f)
- Queens College CUNY [CUNY] (pir9)
- Rutgers University [Rutgers-K] (Ant)
- Seoul [Seoul] (Scai8)
- TNO-TPD / Univ. Twente [TNO] (uttno8)
- University of Iowa [UIowa] (IOWAF)
- University of Massachusetts Amherst [UMass] (INQ)
- University of Maryland [UMaryland] (ume)

Table 1 summarizes the tasks each group participated in and whether or not this is their first year participating.

3.1 Summary of approaches

In this section, we present a one-paragraph description of the techniques used by each of the groups in their TREC-8 experiments. Essentially, this information is just a summary of the final papers. The goal is to enable readers to quickly identify the papers they wish to read and to provide background information to motivate the commentary on the evaluation results which follows. We

were unable to find final papers for IRIT, Rutgers-K, Seoul, and UMass at the time this document was written.

City participated in batch filtering and routing. They focused on measuring the efficiency of their Pliers system for text filtering using an architecture of 16 Pentium II machines in parallel. Their routing system is based on the Okapi TREC-5 strategy of partitioning the training set into two parts: one for term extraction and one for term selection. Their system indexed the training set (FT91-92) in 5 minutes and filtered the test set in less than 11 minutes.

CLARITECH participated in adaptive and batch filtering. For TREC-7, they concentrated on threshold selection and updating. For TREC-8, they added profile-specific updating (as opposed to their former strategy of updating all profiles at the same time after a batch of documents has been processed) and worked to optimize term selection and the profile scoring function. They chose to update each profile individually every n (usually 2 or 4) documents retrieved or after a fixed amount of time has passed. The former update allows the system to quickly take advantage of new training data (and improved performance) while the latter update allows the system to make adjustments to profiles which have retrieved no documents (and had mixed results). They also found that changing the terms in the profile in response to new training data improves performance even though it makes accurate threshold calibration more difficult.

DSO participated in routing. They select terms using the AT&T TREC-6 routing method then assign weights to the terms using a specialized perceptron learning algorithm (dso99rt1). Their second run (dso99rt2) merges the results of perceptron learning with Dynamic Feedback Optimization.

ICDC participated in routing. Each document is initially represented by terms which have a high frequency in the document relative to their frequency in the corpus. In order to reduce correlation among previously selected terms, a Gram-Schmidt orthogonalization technique is used to find the most descriptive term at each step, given the terms already chosen by the model. The learning algorithm is a linear neural network with early stopping to reduce overfitting.

KDD participated in adaptive filtering. They define the contribution of a term to the similarity of a query document pair as the difference between the similarity scores with and without that term. Words with negative scores are prime candidates for query expansion. The basic filtering model is Rocchio relevance feedback with terms also weighted and selected as a function of their contribution as defined above.

Microsoft participated in adaptive filtering. Their basic system uses Okapi term weighting with no query modification or term reweighting based on feedback data. Initial thresholds (probability estimates) are set via logistic regression on a separate training set of documents and topics. The probabilities are a function of the retrieval score, the average score of the top 1% of retrieved documents (also initially estimated from the training set), the maximum possible score, and the length of the query. The thresholds are set low initially to allow more profile learning. Documents are processed in weekly batches and the intercept term in the regression is updated. In general, they find that starting with higher initial thresholds works better.

CUNY participated in all three tasks. For adaptive filtering, they concentrate on threshold updating. Their system defined a starting threshold (T_{hi}), a lower bound threshold (T_{lo}), a precision threshold (G), and a selection rate threshold (SRT). The current threshold T is updated every 2000 documents. If no relevant docs have been seen, the current threshold (T) is decreased if the proportion of documents retrieved is less than SRT, but never less than T_{lo} . Otherwise, the threshold is increased if the precision of the current retrieved set is less than G and decreased if precision is greater than G . For batch Filtering and Routing, CUNY's Pircs system uses genetic algorithms to select and modify multiple profiles. Logistic regression is used for threshold selection.

TNO participated in adaptive filtering. They use a probabilistic retrieval model which assumes

that the user generates the query from an ideal internal representation of a relevant document. The initial thresholds are set to a large multiple of the probability of selecting the query from a random document. Each time a document is selected for a profile, the threshold is adjusted to a value slightly lower than the one which achieves maximum performance on the retrieved document set. The model defines a relevance weight for each term (ranging from 0 to 1) which is updated for new data using the EM algorithm.

Ulowa participated in adaptive filtering. They use a dynamic clustering approach with two acceptance thresholds. The first acceptance threshold is based on similarity to the topic. All documents scoring higher than this threshold are dynamically clustered. When a cluster's similarity first exceeds a second visibility threshold, the most recently retrieved document is sent to the user. Clusters are then tagged as relevant or not relevant based on the assessment of this document, and future documents from relevant clusters are also passed on to the user. A document judged not relevant in a relevant cluster spawns an independent non-relevant cluster.

UMaryland participated in routing. They are interested in using Latent Semantic Indexing (LSI) for collaborative filtering. Routing queries are constructed by relevance feedback and then an LSI dimension reduction is performed in the query space. The goal is to find and reuse common structure in the query set. They submitted one run based on relevance feedback alone (umrqz) and one run with relevance feedback plus LSI (umrlsi). Overall, LSI tended to slightly hurt performance, perhaps because there is minimal topic overlap in the TREC-8 query set.

3.2 Evaluation results

Figures 1-8 summarize the evaluation results for the TREC-8 filtering track. Not all runs appear on all graphs due to scaling problems. Missing runs scored below the lower bound on the vertical axis. There are three different types of graph. Figures 1-2 and 6-7 plot average scaled utility for a range of values for the lower bound. The horizontal axis represents the number of non-relevant documents used to define the lower bound (logarithmic scale). The vertical axis represents the difference in average scaled utility between each system and the baseline (retrieving no documents at all). Therefore, the baseline is represented by the straight horizontal axis marked with index points for the lower bound. Figures 3-5 plot average scaled utility by year for adaptive filtering. Only a small fraction of the documents come from 1991, so it can be ignored. The lower bound in these plots is fixed (the value of s is shown at the bottom of the plot). Figure 8 shows the average uninterpolated precision scores on the left side of the plot. For the right side, a system's score is replaced by its rank with respect to that topic, and these ranks are averaged across topics. The results are then rescaled to cover the same range as the raw scores.

The reader will note that there are no plots for non-linear utility. Only two groups submitted runs optimized for these measures, one by CLARITECH for NF1 and adaptive filtering, one by IRIT for NF1 and batch filtering. Therefore, there are no meaningful comparisons that can be made. We had hoped that the non-linear utility functions would do a better job of equalizing the size of the retrieved sets. However, a comparison of the variation in the size of the retrieved set between CLARITECH's NF1 run and their four LF1 runs shows that there is no significant difference. Given the lack of interest in the non-linear utility functions, it is unlikely that this experiment will continue next year. Several groups expressed the sentiment that the non-linear utility functions do not represent a good user model for filtering.

Figure 1 shows average scaled utility for the LF1 function and the adaptive filtering task. This plot demonstrates that the LF1 function is a very challenging standard, as none of the systems manage to beat the baseline. Most of the lines do not cross, indicating that relative system performance changes very little as a function of the lower bound. The convergence of lines as the lower

bound increases is a natural phenomenon caused by the fact that the possible range of scores is much larger, therefore the actual scaled scores will cluster in a smaller part of the space. This pattern will be observed in all plots of this type. The two most successful runs (IOWA-2 and TNO-1p) owe their success to their extremely conservative document selection strategies. Figure 2 shows average scaled utility for the LF2 function and the adaptive filtering task. The task here is simpler, and five systems consistently beat the baseline.

Figures 3 and 4 show average scaled utility for the LF1 function and adaptive filtering broken down by year. The difference between the two figures is the scale of the vertical axis. In addition, the TNO and CLARITECH runs were averaged in Figure 3 to avoid crowding the display at the larger scale. All systems start below the baseline in 1992, due to the need to calibrate thresholds by retrieving a few documents with little prior information about the odds that they are relevant. It is encouraging to see that by 1993 six different runs are above the baseline and this pattern continues into 1994. However, there are striking differences in system behavior in these plots. We can distinguish three different patterns in this plot. CL99afL1(b,c) and uttno8lf1 improve gradually and consistently over time. CL99afL1(a,d) and uttno8lf1(f,p) improve significantly in 1993 and then fall back slightly in 1994. Microsoft (ok8f), and to a lesser extent UMass (INQ) and UIowa, do more poorly in 1993 but recover nicely in 1994. This indicates that learning behavior differs substantially from system to system over time.

Figure 5 shows average scaled utility for the LF2 function and adaptive filtering broken down by year. We see similar patterns to Figure 4, except that most systems are well above the baseline in 1993 and 1994. Note that the utility scores were scaled independently by year. This means that the scores are not additive, i.e. you can't add up the scores for a system in Figure 5 and get the score in Figure 4 at the same lower bound. This is because the upper bound is different for each plot. In hindsight, it might have made more sense to scale once and then divide up the score by year.

Figures 6 and 7 show average scaled utility for the LF1 and LF2 functions respectively and batch filtering. Almost all systems score well above the baseline. Figure 8 shows the average uninterpolated precision and average scaled rank for routing systems.

3.3 Utility vs. Other Measures

The evaluation results for adaptive and batch filtering presented in the previous section are all based on scaled utility. In order to give a broader view of system performance, we look at a number of alternative measures in Table 2 and Table 3 below. Scores are tabulated for the best run from each of the five groups with the highest average scaled utility ($U(s) = -100$). The measures include average size of the retrieved set, micro-averaged precision (precision of retrieved set as a whole), total number of relevant documents retrieved, and the number of topics with utility score greater than zero. We use micro-averaging for precision to avoid the problem of empty retrieved sets. It is clear from the results that scaled utility hides a lot of important differences in system behavior.

The most striking pattern for LF1 is that scaled utility is inversely correlated with average set size and the total number of relevant documents. In other words, retrieving more documents and more relevant documents leads to worse performance. Even more surprising, uttnolflp and CL99afL1d have a higher precision and a higher recall than IowaF992 but receive a lower utility score! This is because the precision threshold for LF1 is 0.4, and all systems retrieve with a precision below this level. When a system fails to meet this threshold, each new document has a negative expected utility, and retrieving more documents lowers the overall utility score. Therefore, higher precision *and* higher recall does not always translate into better performance when evaluation is based on utility. This particular property runs counter to most intuitive notions in information

Run Name	u_{50}^* LF1	Average Set Size	Micro Avg Precision	Total Rel.	# Topics LF1 > 0
Baseline	0.0	0.0	—	0	0
IowaF992	-0.012	3.5	0.32	57	10
uttnolf1p	-0.015	5.7	0.36	102	11
CL99afL1d	-0.025	10.0	0.36	180	11
INQ610	-0.071	17.4	0.31	266	12
ok8f311	-0.153	23.3	0.25	295	14

Table 2: Comparison of LF1 to other measures.

Run Name	u_{100}^* LF2	Average Set Size	Micro Avg Precision	Total Rel.	# Topics LF2 > 0
uttnolf2f	0.034	23.2	0.32	372	20
CL99afL2	0.025	11.3	0.35	197	17
INQ612	0.013	17.4	0.31	266	18
IowaF991	0.007	4.3	0.34	74	13
Baseline	0.0	0.0	—	0	0
ok8f321	-0.011	29.2	0.27	398	23

Table 3: Comparison of LF2 to other measures.

retrieval and may help explain why many people are uncomfortable with the utility measure.

For LF2, system behavior is more regular. Generally, the systems that retrieve more documents with a higher precision score better. In this case, all the top-ranked systems exceed the LF2 precision threshold of 0.25. It is sobering to note that even for the easier LF2 function, the best systems retrieve with positive utility on fewer than half the topics. For LF1, the best systems beat the baseline on fewer than 30% of the topics!

It is also interesting to note that with the exception of TNO, there is much more similarity (in terms of set size, precision, and number relevant) between runs from the same system for LF1 and LF2 than between different systems for the same utility function. This pattern is exemplified by the success of INQ610, which appears to be the same run submitted for both utility measures! One possible explanation is that many topics are highly insensitive to the choice of retrieval threshold and that differences in profile construction are more important for changing the size of the retrieved set. Most likely, this reflects the fact that best performance for LF1 and LF2 for most systems on most topics is to retrieve no documents, and a wide range of thresholds achieve this outcome.

4 General Commentary

Following the progression of system performance from TREC-7 to TREC-8 (or lack thereof!), it is becoming increasingly clear that the adaptive filtering task is too hard. Once again, no system performed better than the baseline over all three years for the LF1 utility function. In other words, it is better not to retrieve any documents according to this standard. Furthermore, systems with higher precision and higher recall get a lower score, because they are unable to retrieve with high enough precision. While systems are clearly ahead of the baseline for LF2, gains are concentrated in fewer than half the topics. Looking at Figures 3-5, we realize that the situation is not as bad as it first appears. Most systems suffer during the start-up phase but go on to beat the baseline quite substantially in 1993 and 1994. However, the current utility functions (particularly LF1) simply

penalize too strongly against retrieving non-relevant documents early to allow systems to develop an overall winning strategy.

This suggests a clear road for improvement in TREC-9. Next year, we propose to supplement the topic statement with one or two positive training examples in order to give systems more of a head start. In addition, we will find a topic set with a larger number of relevant documents. The current TREC tasks favor highly conservative filtering strategies which creates an environment where there is little opportunity for adaptive learning. In addition, there is some interest in working with evaluation measures based on different user models, such as: find n relevant documents as fast as possible or return no more than k documents per unit time. We are strongly considering moving to a text categorization test collection for TREC-9, to give us more flexibility in topic selection and to reduce the assessment load for NIST. This will introduce new challenges, such as defining an acceptable topic statement from category labels that may not be well defined.

When looking over the system reports, several clear patterns emerge. As suggested in our commentary last year, it is important to take advantage of new training data as quickly as possible. For TREC-7, most systems used batch updating, learning new profiles and threshold simultaneously for all topics every k documents filtered. In TREC-8, the most successful systems are now updating their profiles independently with a much smaller batch ratio (usually 1 or 2 documents). Systems like Okapi and Pircs, which still use large batches, have much more trouble learning rapidly in the first year, a crucial period for determining overall performance. Differences in updating strategies may explain two of the dominant learning behaviors found in Figures 3 and 4. These systems gain more training data, which enables them to make more informed decisions later on, but not enough to overcome the cost of all the irrelevant material passed on to the user. Given that people tend to expect good results as soon as possible, this is not likely to be a winning strategy in a real filtering system.

As in TREC-7, most groups concentrate on optimizing their system's adaptive threshold setting ability, rather than changing the terms or the term weights in the profile. Once again, the TREC-8 task encourages highly conservative filtering strategies, which limits the ability of systems to adaptively learn better profiles. Nonetheless, groups which run comparisons with and without profile updating, such as CLARITECH and TNO/Twente, find that profile updating does improve performance. We hope to encourage more work in this area by revising the task for next year. Overall, the TREC filtering track continues to grow and prosper and we look forward to welcoming new participants next year.

Acknowledgements We give our thanks to all the people who have contributed to the development of the TREC filtering track over the years, in particular David Lewis, Karen Sparck Jones, Chris Buckley, Paul Kantor, Ellen Voorhees, R.W. Hutchinson, Djoerd Hiemstra, the TREC program committee, and the team at NIST.

References

- [1] I.J. Good. The Decision Theory Approach to the Analysis of Information Retrieval Systems. *Information Storage and Retrieval Systems*, 3:31–34, 1967.
- [2] David A. Hull. The TREC-6 Filtering Track: Description and Analysis. In *The 6th Text Retrieval Conference (TREC-6)*, NIST SP 500-240, pages 45–68, 1998.
- [3] David A. Hull. The TREC-7 Filtering Track: Description and Analysis. In *The 7th Text Retrieval Conference (TREC-7)*, NIST SP 500-242, pages 33–56, 1999.

- [4] David Lewis. The TREC-4 Filtering Track. In *The 4th Text Retrieval Conference (TREC-4)*, NIST SP 500-236, pages 165–180, 1996.
- [5] David Lewis. The TREC-5 Filtering Track. In *The 5th Text Retrieval Conference (TREC-5)*, NIST SP 500-238, pages 75–96, 1997.

Figure 1 - Adaptive Filtering: Scaled LF1 Utility (92-94)

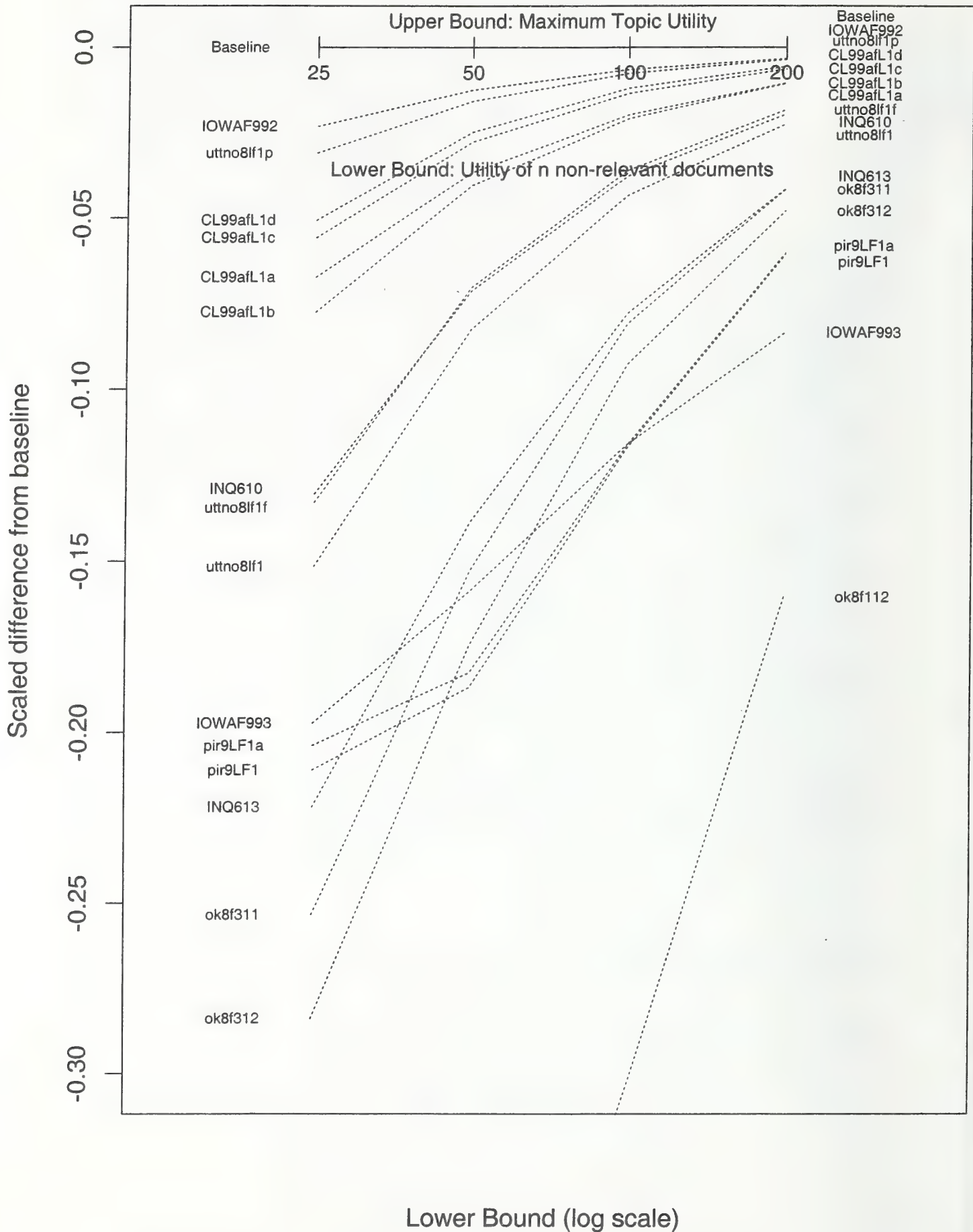


Figure 2 - Adaptive Filtering: Scaled LF2 Utility (92-94)

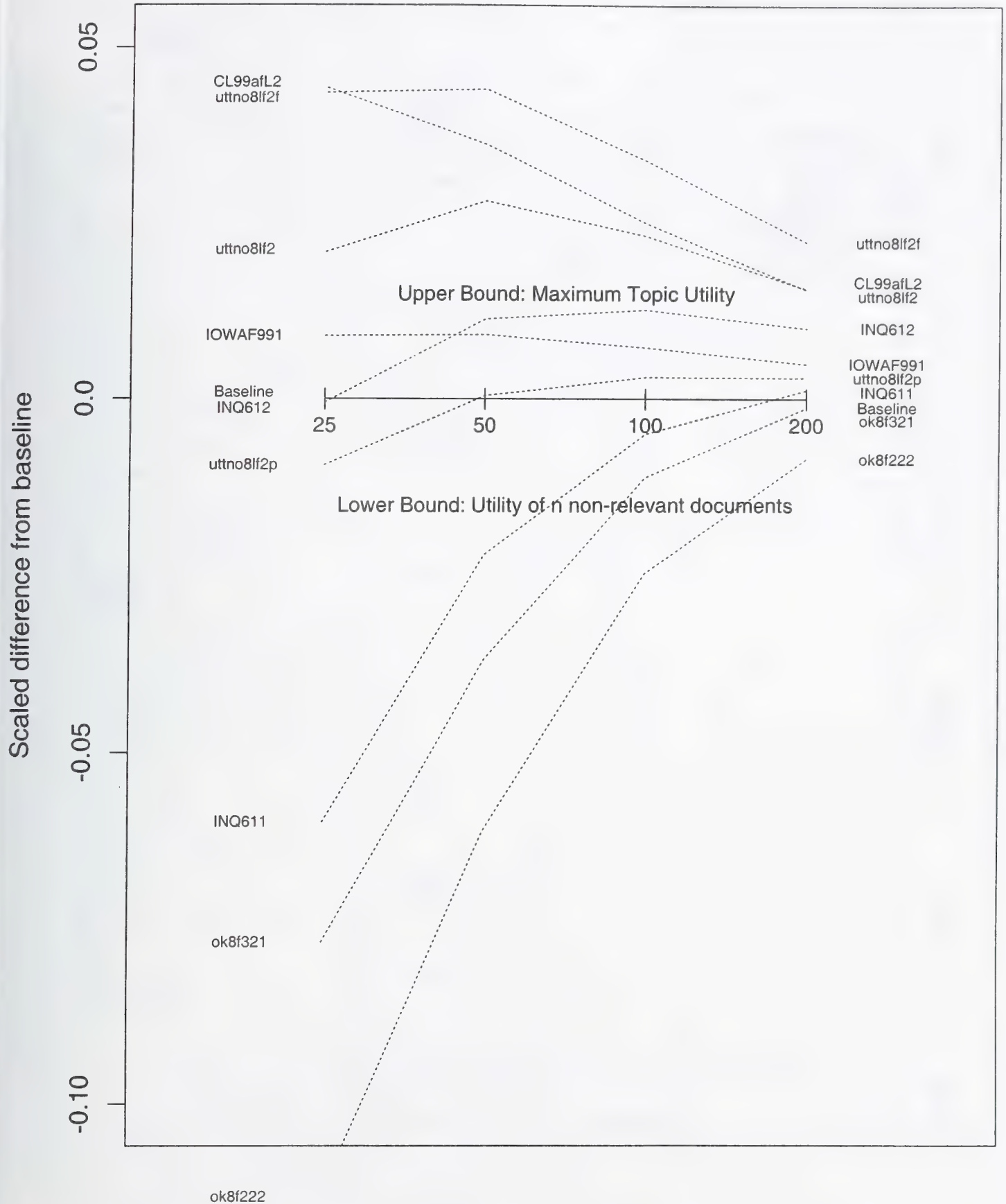


Figure 3 - Adaptive Filtering by year (Scaled LF1 Utility)

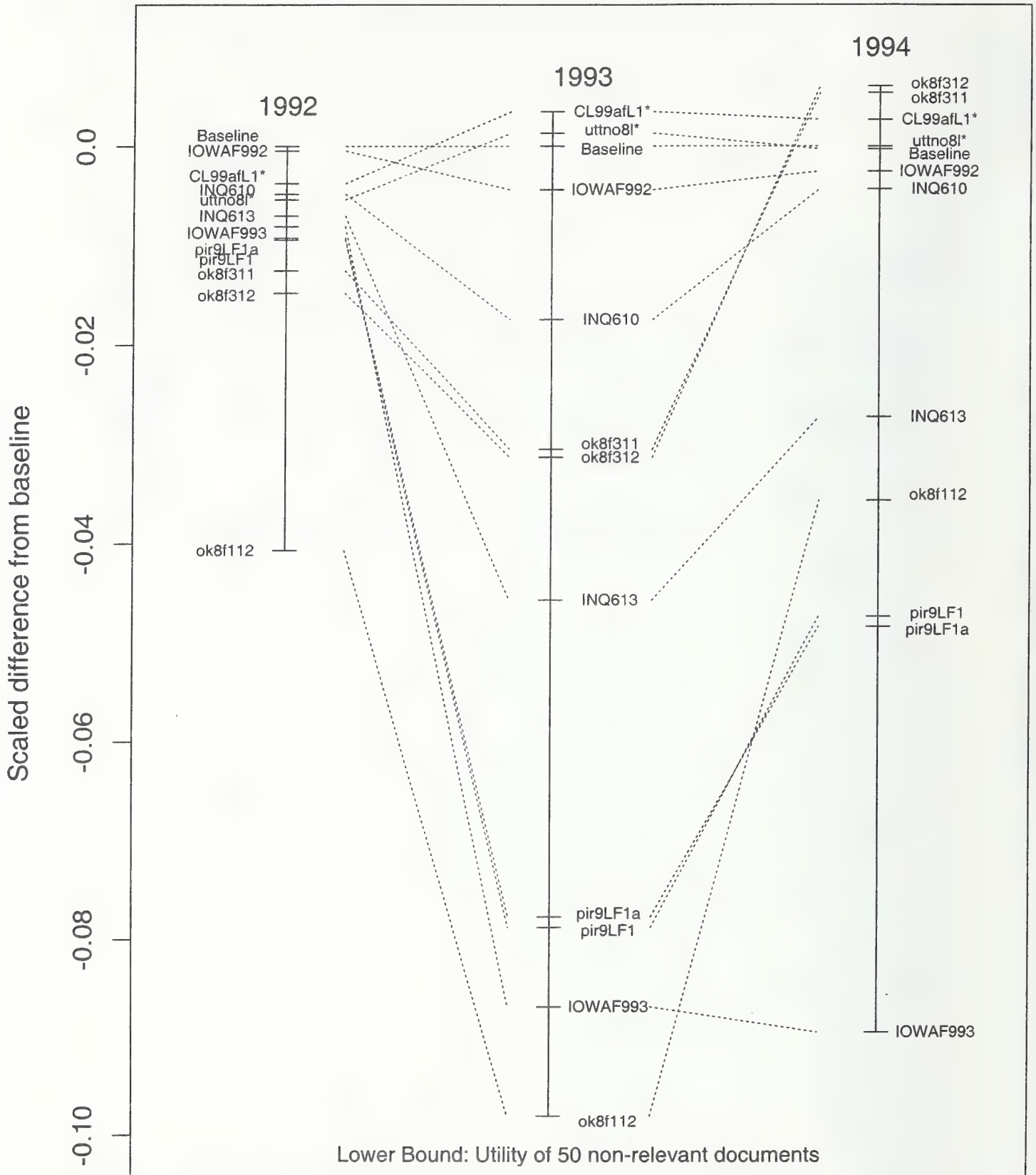


Figure 4 - Adaptive Filtering by year (Scaled LF1 Utility)

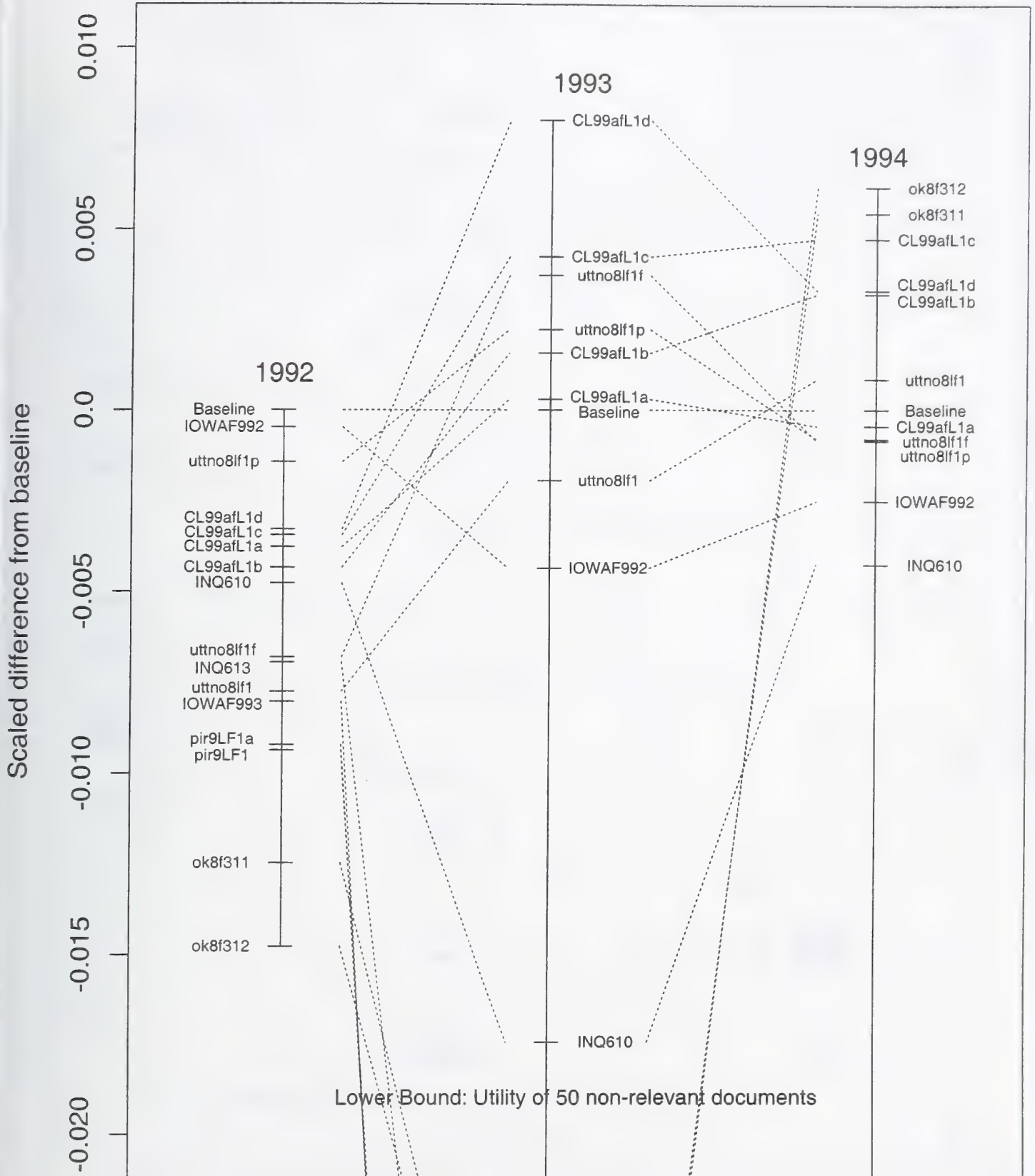


Figure 5 - Adaptive Filtering by year (Scaled LF2 Utility)

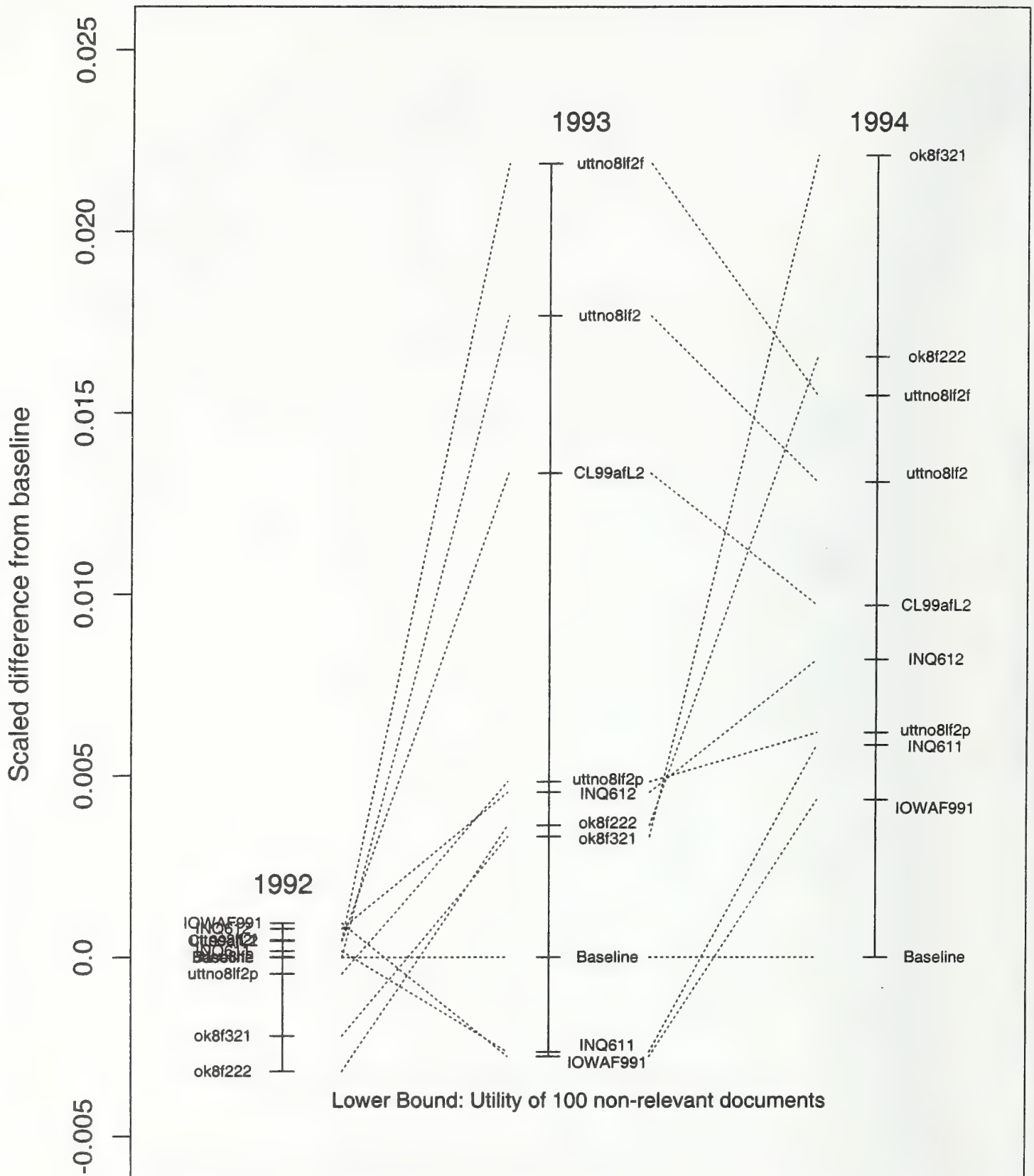


Figure 6 - Batch Filtering: Scaled LF1 Utility (93-94)

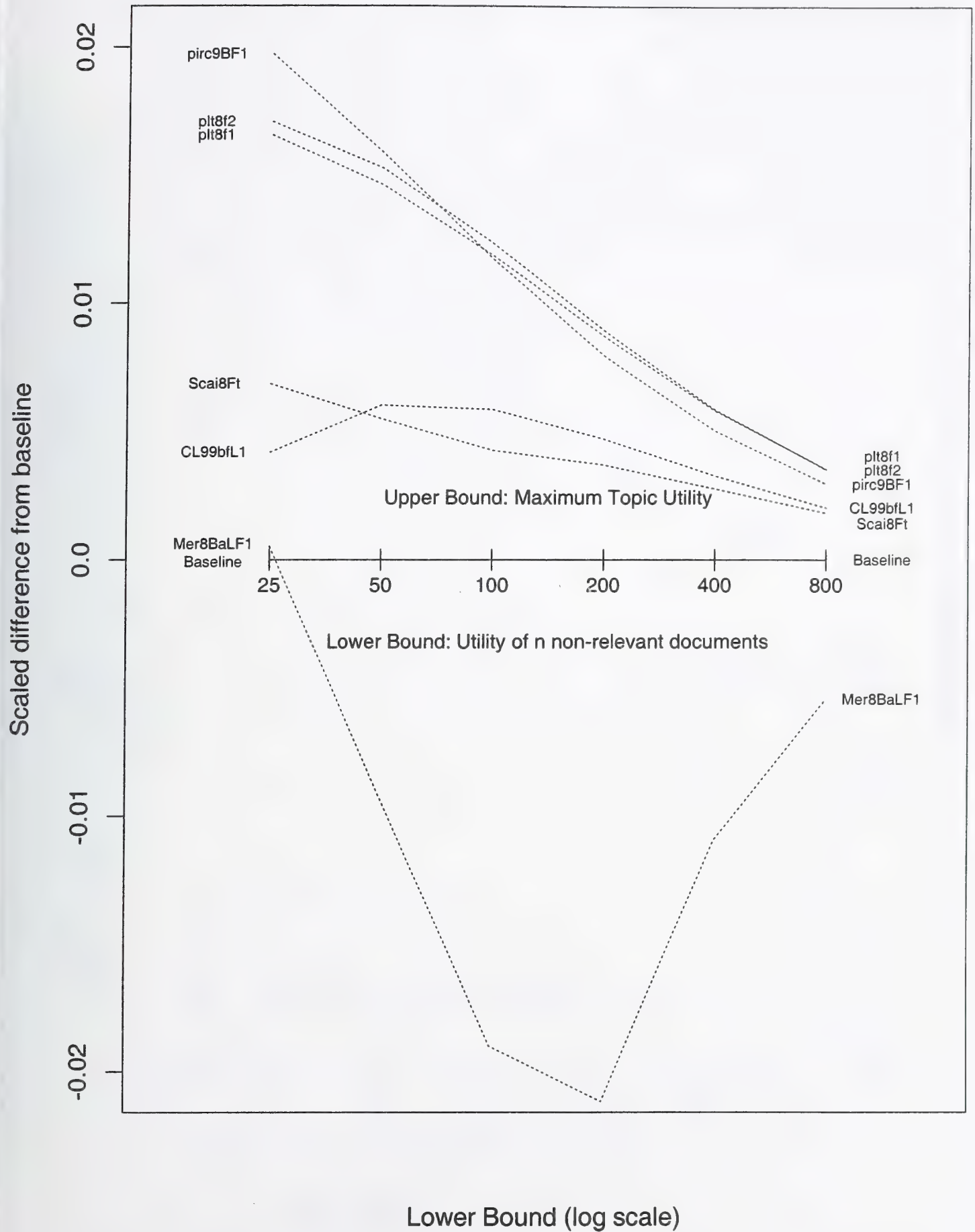


Figure 7 - Batch Filtering: Scaled LF2 Utility (93-94)

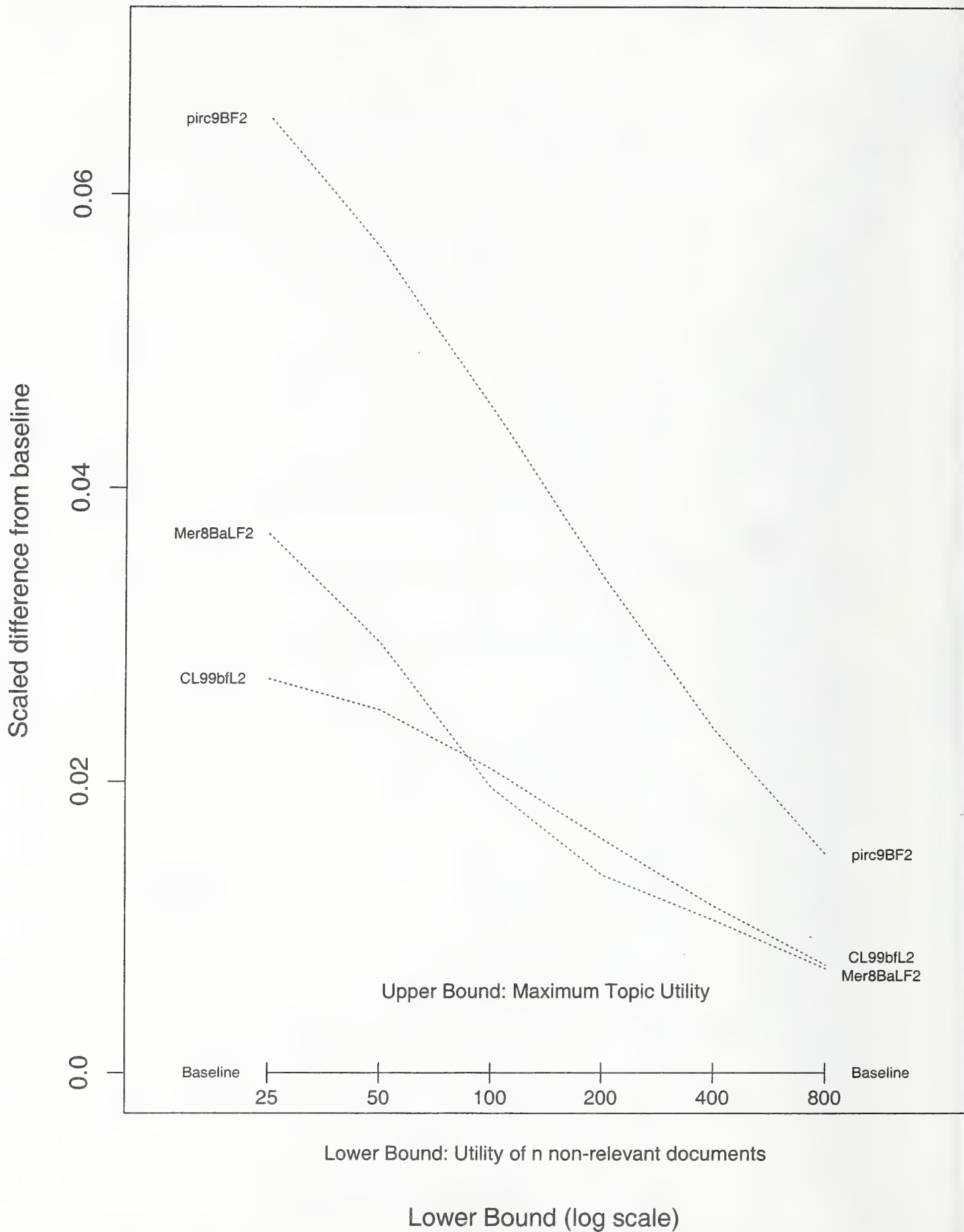
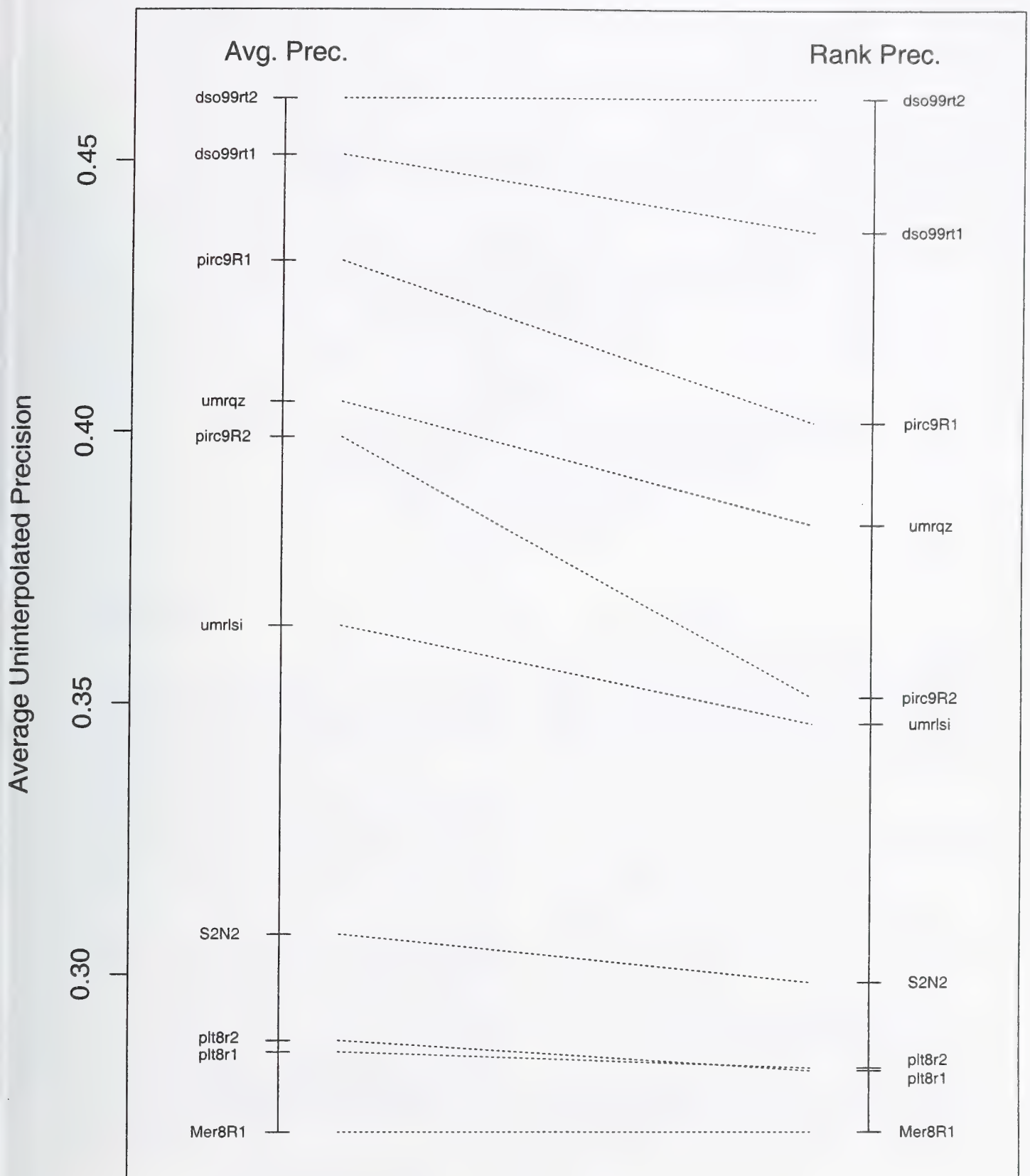


Figure 8 - Routing: Averaged vs. Ranked Precision



TREC-8 Interactive Track Report

William Hersh
hersh@ohsu.edu

Division of Medical Informatics & Outcomes Research
Oregon Health Sciences University
Portland, OR 97201, USA

Paul Over
over@nist.gov

Natural Language Processing and Information Retrieval Group
Information Access and User Interfaces Division
National Institute of Standards and Technology
Gaithersburg, MD 20899, USA

October 25, 2000

Abstract

This report is an introduction to the work of the TREC-8 Interactive Track with its goal of investigating interactive information retrieval by examining the process as well as the results.

Seven research groups ran a total of 14 interactive information retrieval (IR) system variants on a shared problem: a question-answering task, six statements of information need, and a collection of 210,158 articles from the Financial Times of London 1991-1994.

This report summarizes the shared experimental framework, which for TREC-8 was designed to support analysis and comparison of system performance only within sites. The report refers the reader to separate discussions of the experiments performed by each participating group — their hypotheses, experimental systems, and results. The papers from each of the participating groups and the raw and evaluated results are available via the TREC home page (trec.nist.gov).

1 Introduction

For TREC-8 the high-level goal of the Interactive Track remained the investigation of searching as an interactive task by examining the process as well as the outcome. To this end a common experimental framework was designed with the following features:

- an interactive search task
- 6 topics — brief statements of information need
- a document collection to be searched
- a required set of searcher questionnaires
- 5 classes of data to be collected at each site and submitted to NIST
- 3 summary measures to be calculated by NIST for use by participating research groups

The framework allowed groups to estimate the effect of their experimental manipulation free and clear of the main (additive) effects of searcher and topic

Table 1: Participating research groups, their systems, and the number of searches performed on each.

Group	Searches
New Mexico State University at Las Cruces	72
Oregon Health Sciences University	144
Royal Melbourne Institute of Technology / CSIRO	144
Rutgers University	216
Sheffield University	144
University of California at Berkeley	72
University of North Carolina at Chapel Hill	144

and it was designed to reduce the effect of interactions, e.g., searcher with topic, topic with system, etc.

In TREC-8 the emphasis was on each group's exploration of different approaches to supporting the common searcher task and understanding the reasons for the results they get. No formal coordination of hypotheses or comparison of systems across sites was planned, but groups were encouraged to seek out and exploit synergies. Some groups designed/tailored their systems to optimize performance on the task; others simply used the task to exercise their system(s). Table 1 lists the research groups that took part and the total number of searches performed as part of their experiment. The issues addressed by each team are discussed in section 3.

2 Method

2.1 Participants

Each research group selected its own experimental participants, known here as "searchers." There was only one restriction: no searcher could have previously used either the control system or the experimental system. Additional restrictions were judged impractical given the difficulty of finding searchers. A minimum of twelve searchers was required, but the experimental design allowed for the addition of more in groups of four and additions were encouraged. Standard demographic data about each searcher were collected by each site and some sites administered additional tests.

2.2 Apparatus

IR systems

In addition to running its experimental system(s), each participating site chose a control system appropriate to the local research goals.

Computing resources

Each participating group was responsible for its own computing resources adequate to run both the control and experimental systems and collect the data required for its own experiments and for submission to NIST. The control and the experimental systems were to be provided with equal computing resources within a site but not necessarily the same as those provided at other sites.

Topics

Six of the 50 topics created by NIST for the TREC-8 adhoc task were selected and modified for use in the interactive track by adding a section called "Instances" and removing the "Narrative." The six topics were entitled as follows:

- 408i tropical storms
- 414i Cuba, sugar, imports
- 428i declining birth rates

- 431i robotic technology
- 438i tourism, increase
- 446i tourists, violence

Each of the six topics described a need for information of a particular type. Contained within the documents of the collection to be searched were multiple distinct examples or instances of the needed information. Here is an example interactive topic.

Number: 408i

Title: tropical storms

Description:

What tropical storms (hurricanes and typhoons) have caused property damage and/or loss of life?

Instances:

In the time allotted, please find as many DIFFERENT storms of the sort described above as you can. Please save at least one document for EACH such DIFFERENT storm. If one document discusses several such storms, then you need not save other documents that repeat those, since your goal is to identify as many DIFFERENT storms of the sort described above as possible.

Searcher task

The task of the interactive searcher was to save documents, which, taken together, contained as many different instances as possible of the type of information the topic expressed a need for — within a 20 minute time limit.

Searchers were encouraged to avoid saving documents which contribute no instances beyond those in documents already saved, but there was no scoring penalty for saving such documents and searchers were to be told that.

Table 2: Basic 2x2 Latin square on which evaluation is based.

Searchers	System, Topic combinations
S1	E, Tx C, Ty
S2	C, Ty E, Tx

Document collection

The collection of documents to be searched was the Financial Times of London 1991-1994 collection (part of the TREC-8 adhoc collection). This collection contains 210,158 documents (articles) totaling 564 megabytes. The median number of terms per document is 316 and the mean is 412.7.

2.3 Procedure

Each searcher performed six searches on the document collection using the six interactive track topics in a pseudo-random order. Each searcher performed 3 searches on one of the site's systems and then 3 on the other to avoid the extra cognitive load of switching systems with each search. Instructions on the task preceded all searching and a system tutorial preceded the first use of each system. In addition, each searcher was asked to complete a questionnaire, prior to all searching, after each search, after the last search on a given system, and after all searching was complete. The detailed experimental design determined the pseudo-random order in which each searcher used the systems (experimental and control) and topics.

The minimal 12-searcher-by-6-topic matrix can be rearranged and seen as 18 2-searcher-by-2-topic Latin squares. Each 2-by-2 square has the form shown in Table 2 and has the property that the "treatment effect," here $E - C$, the control-adjusted response, can be estimated free and clear of the main (additive) effects of searcher and topic. Participant and topic are treated statistically as blocking factors. This means that even in the presence of the anticipated differ-

Table 3: Half the minimal 8-searcher-by-8-topic matrix as run.

Searchers	System, Topic combinations (in example order as seen by searchers)					
S1	E,T6	E,T1	E,T2	C,T3	C,T4	C,T5
S2	C,T1	C,T2	C,T3	E,T4	E,T5	E,T6
S3	C,T2	C,T3	C,T4	E,T5	E,T6	E,T1
S4	C,T3	C,T4	C,T5	E,T6	E,T1	E,T2
S5	E,T4	E,T5	E,T6	C,T1	C,T2	C,T3
S6	E,T5	E,T6	E,T1	C,T2	C,T3	C,T4
S7	C,T6	C,T1	C,T2	E,T3	E,T4	E,T5
S8	E,T1	E,T2	E,T3	C,T4	C,T5	C,T6
S9	E,T2	E,T3	E,T4	C,T5	C,T6	C,T1
S10	E,T3	E,T4	E,T5	C,T6	C,T1	C,T2
S11	C,T4	C,T5	C,T6	E,T1	E,T2	E,T3
S12	C,T5	C,T6	C,T1	E,T2	E,T3	E,T4

ences between searchers and topics, the designs provided estimates of $E - C$ that were not contaminated by these differences.

However, the estimate of $E - C$ would be contaminated by the presence of an interaction between topic and searcher. Therefore, we replicated the 2x2 Latin square 6x3 times to get the minimal 12x6 design for each site. The contaminating effect of the topic by searcher interaction was reduced by averaging the eighteen estimates of $E - C$ that are available, one for each 2x2 Latin square. This is analogous to averaging replicate measurements of a single quantity in order to reduce the measurement uncertainty. Each 2-by-2 square yields 1 within-searcher estimate of the $E - C$ difference for a total of 18 such estimates for each 12-searcher-by-6-topic matrix.

In resolving experimental design questions not covered here (e.g., scheduling of tutorials and searches, etc.), participating sites were asked to minimize the differences between the conditions under which a given searcher used the control and those under which he or she used the experimental system.

2.4 Data submitted to NIST

Six sorts of data were collected for evaluation/analysis (for all searches unless otherwise specified) and are available from the

Table 4: Results by topic.

Topic	Mean instance recall across all searcher-systems	Mean instance precision across all searcher-systems	Number of instances identified by NIST
408i	0.326	0.777	24
414i	0.532	0.660	12
428i	0.306	0.667	26
431i	0.329	0.821	40
438i	0.172	0.734	56
446i	0.227	0.517	16

TREC-8 Interactive Track web page ([www-nlpir.nist.gov/projects/t8i/t8i.html](http://www.nlpir.nist.gov/projects/t8i/t8i.html)).

- sparse-format data — list of documents saved and the elapsed clock time for each search
- rich-format data — searcher input and significant events in the course of the interaction and their timing
- searcher questionnaires on background, user satisfaction, etc.
- a full narrative description of one interactive session for topic 408i
- any further guidance or refinement of the task specification given to the searchers

Only the sparse-format data were evaluated at NIST to produce a triple for each search: instance precision and recall (these as defined in the next section) and elapsed clock time.

2.5 Evaluation of the sparse-format data submitted to NIST

Evaluation by NIST of the sparse-format data proceeded as follows. For each topic, a pool was formed containing the unique documents saved by at least one searcher for that topic regardless of site.

For each topic, the NIST assessor, normally the topic author, was asked to:

1. Read the topic carefully.
2. Read each of the documents from the pool for that topic and gradually:
 - (a) Create a list of the instances found somewhere in the documents
 - (b) Select and record a short phrase describing each instance found
 - (c) Determine which documents contain which instances
 - (d) Bracket each instance in the text of the document in which it was found

Then for each search (by a given searcher for a given topic at a given site), NIST used the submitted list of selected documents and the assessor's instance-document mapping for the topic to calculate:

- the fraction of total instances (as determined by the assessor) for the topic that are covered by the submitted documents (i.e., instance recall)
- the fraction of the submitted documents which contain one or more instances (i.e., instance precision)

The third measure, elapsed clock time, was taken directly from the submitted results for each search.

3 Results and Discussion

The mean results by topic are presented here in Table 4. For TREC-8, topic presentation sequence was randomized for each searcher.

A summary of each group's results (instance recall by site and condition) is shown in Table 5. Comparison of systems across sites is not supported by the experimental design, so comparisons presented here are between systems within a given site. A general theme running through the results was that there was little difference between each group's experimental and control systems. Whether it was New Mexico State University's document summarization approach or

Table 5: Instance recall by site and condition.

Site	Condition	Instance recall
NMSU	Added document summaries	0.44
	Baseline full text	0.40
OHSU	Okapi weighting	0.38
	Baseline tf*idf weighting	0.33
RMIT/CSIRO	Added categorization interface	0.27
	Baseline document list interface	0.31
Rutgers	Relevance feedback	0.26
	Local context analysis	0.24
Sheffield	With relevance feedback	0.35
	Without relevance feedback	0.39
Berkeley	Enhanced Cheshire interface	0.38
	Baseline ZPRISE Interface	0.41
UNC	Passage-level retrieval feedback	0.23
	Document-level retrieval feedback	0.28

the use of the relevance feedback by Sheffield University, users showed little difference across systems, many of which contained features shown to be effective in non-interactive experiments in the past. A generalization can be made that these techniques, such as relevance feedback, Okapi weighting, document summarization, and greater control over search terms and Boolean operators, do not show benefit in the instance recall task.

There are two possible explanations for this. Either there really is no difference or these experiments lack the research design or statistical power to detect a difference. Only further research, including the study other types of search tasks and larger numbers of queries, will resolve this.

The actual results obtained by each group are summarized in the following paragraphs. For more details the reader is directed to the site reports in these proceedings or on the TREC web site (trec.nist.gov).

- New Mexico State University looked at whether users could find relevant information with a user interface for viewing retrieval results that showed

query term occurrence and distribution along with extracted names of people and locations shown in document surrogate lists and summaries. Their results showed no difference in instance recall between the two systems. However, 11 of 12 users reported that they liked the summary display better than the full text control. However, this preference did not match performance. Of note was that users viewed more documents in the full text condition.

- Oregon Health Sciences University used the interactive track to assess whether batch and user evaluations give the same results. Batch experiments with TREC-6/7 data showed substantial differences for various weighting schemes, and particular benefit for Okapi. User experiments showed no such comparable benefit. For more information see the site report in these proceedings (Hersh et al., 2000).
- Royal Melbourne Institute of Technology-CSIRO tested the hypothesis that by allowing the user control over the organization of the information, and the selection of documents using the organization, the user would find a better set of documents to view, and hence achieve a better coverage of aspects. Their control system featured three windows with a list of document titles, one document displayed, and a saved instances window. The experimental system replaced the list of document titles with a window containing a list of categories and documents clustered therein. The categories were derived from WordNet. Results showed that using the categorized interface, users read more documents, saved the same number of documents, and saved more aspects, but with less accuracy. User satisfaction did favor the categorized system. For more information see the site report in these proceedings (Fuller et al., 2000).
- Rutgers University compared two different techniques for supporting query reformulation by term suggestion in interactive IR: user-controlled relevance feedback (RF) and system-controlled Local Context Analysis (LCA). Their results

showed that LCA did not perform better, but was easier for the user. Effectiveness and usability were the same for each system. In LCA mode, more terms were suggested than in RF and more suggested terms were used in the queries, which were equally long in both systems. Thus users had to do less (cognitive) work in LCA. The authors speculated that if LCA terms were "better," then maybe the approach would be more effective, usable, and preferred. For more information see the site report in these proceedings (Belkin et al., 2000).

- Sheffield University focused on searching behavior and user perception of an experimental retrieval task assessing the impact of document ranking, best-passage retrieval, and a query expansion facility. The experimental setting used two versions of Okapi, one with relevance feedback and one without. Their findings showed that while user outcomes were the same, search confidence was positively associated with the number of instances retrieved. For more information see the site report in these proceedings (Beaulieu, Fowkes, Alemayehu, & Sanderson, 2000).
- University of California, Berkeley, assessed new features added to its Cheshire II experimental system, in particular the Boolean NOT capability and new ways for navigating results and selecting relevant items. Users achieved the same instance recall as they did with the previous system. For more information see the site report in these proceedings (Larson, 2000).
- University of North Carolina found no difference among various levels of relevance feedback. For more information see the site report in these proceedings (Yang & Maglaughlin, 2000).

These results place an imperative on continued user-oriented evaluation. While non-interactive evaluation will continue to have its role, such as in assessing the feasibility of new algorithms and approaches and the parameterization of system features, interactive experiments must verify that new system advances can be used with their intended beneficiaries,

real users. Just because users and user studies are unpredictable as well as resource-consuming, this does not mean we should avoid them.

Since the interactive track has focused on the instance recall task for three years running, a growing consensus of participating groups prefer to assess different retrieval tasks and documents. Next year's track will likely move to more of a question-answering approach using data from the Web track. The participants also hope to explore specific aspects of the interactive retrieval task. For example, future experiments might decompose the overall task into pieces, such as query composition or document selection. Likewise, there is a desire to base experiments on sound underlying models, such as those of the user, the task, and the typology of information needs. Future discussion will ensue on the track list-serv (trec-int@ohsu.edu).

4 Authors' note

The design of the TREC-8 Interactive Track matrix experiment grew out of the efforts of the many people who contributed to the discussion of ends and means on the track discussion list and through other channels.

References

- Beaulieu, M., Fowkes, H., Alemayehu, N., & Sanderson, M. (2000). Interactive Okapi at Sheffield - TREC-8. In E. M. Voorhees & D. K. Harman (Eds.), *The Eighth Text REtrieval Conference (TREC-8)*. Gaithersburg, MD, USA.
- Belkin, N. J., Head, J., Jegn, J., Kelly, D., Lin, S., Park, S. Y., Cool, C., Savage-Knepshield, P., & Sikora, C. (2000). Relevance Feedback versus Local Context Analysis as Term Suggestion Devices: Rutgers' TREC-8 Interactive Track Experience. In E. M. Voorhees & D. K. Harman (Eds.), *The Eighth Text REtrieval Conference (TREC-8)*. Gaithersburg, MD, USA.
- Fuller, M., Kaszkiel, M., Kimberley, S., Zobel, C., Justinand Ng, Wilkinson, R., & Wu, M. (2000). The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8. In E. M. Voorhees & D. K. Harman (Eds.), *The Eighth Text REtrieval Conference (TREC-8)*. Gaithersburg, MD, USA.
- Hersh, W., Turpin, A., Price, S., Kraemer, D., Chan, B., Sacherek, L., & Olson, D. (2000). Do Batch and User Evaluations Give the Same Results?: An Analysis from the TREC-8 Interactive Track. In E. M. Voorhees & D. K. Harman (Eds.), *The Eighth Text REtrieval Conference (TREC-8)*. Gaithersburg, MD, USA.
- Larson, R. R. (2000). Berkeley's TREC-8 Interactive Track Entry: Cheshire and Zprise. In E. M. Voorhees & D. K. Harman (Eds.), *The Eighth Text REtrieval Conference (TREC-8)*. Gaithersburg, MD, USA.
- Yang, K., & Maglaughlin, K. L. (2000). IRIS at TREC-8. In E. M. Voorhees & D. K. Harman (Eds.), *The Eighth Text REtrieval Conference (TREC-8)*. Gaithersburg, MD, USA.

5 Appendix: Instructions to be given to each searcher

The following introductory instructions are to be given once to each searcher before the first search:

Imagine that you have just returned from a visit to your doctor during which it was discovered that you are suffering from high blood pressure. The doctor suggests that you take a new experimental drug, but you wonder what alternative treatments are currently available. You decide to investigate the literature on your own to satisfy your need for information about what different alternatives are available to you for high blood pressure treatment. You really need only one document for each of the different treatments for high blood pressure.

You find and save a single document that lists four treatment drugs. Then you find and save another two documents that each discusses a separate alternative treatment: one that discusses the use of calcium and one that talks about regular exercise. You've run out of time and stop your search. In all, you have identified six different instances of alternative treatments in three documents.

In this experiment, you will face a similar task. You will be presented with several descriptions of needed information on a number of topics. In each case there can be multiple examples or instances of the type of information that's needed.

We would like you to identify as many different instances as you can of the needed information for each topic that will be presented to you - as many as you can in the 20 minutes you will be given to search. Please save one document for EACH DIFFERENT instance of the needed information that you identify. If you save one document that contains several instances, try not to save additional documents that contain ONLY those instances. However, you will not be penalized if you save documents unnecessarily.

As you identify an instance of the needed information, please keep track of which instances you have found: write down a word or short phrase to identify the instance, or if the system provides a facility to keep track of instances—use it.

Carefully read each topic to understand the type of information needed. This will vary from topic to topic. On one topic you may be looking for instances of a certain kind of event. On another you may be searching for examples of certain sorts of people, places, or things.

Do you have any questions about

- what we mean by instances of needed information,
- the way in which you are to save nonre-

dundant documents for each instance?

The TREC-8 Query Track

Chris Buckley and Janet Walz

Sabir Research, Inc
{chrisb,jwalz}@sabir.com

1 Introduction

The Query Track in TREC-8 is a bit different from all the other tracks. It is a co-operative effort among the participating groups to look at the issue of “query variability.”

The evaluation averages presented in a typical system evaluation task, such as the TREC Ad-Hoc Task, conceal a tremendous variability of system performance across topics/queries. No system can possibly perform equally well on all topics: some information needs (expressed by topics) are harder than others. But what is quite surprising, especially to people just starting to look at IR, is the large variability in system performance across topics as compared to other systems. In a typical TREC task, no system is the best for all the topics in the task. It is extremely rare for any system to be above average for all the topics. Instead, the best system is normally above average for most of the topics, and best for maybe 5%-10% of the topics. It very often happens that quite below-average systems are also best for 5%-10% of the topics, but do poorly on the other topics. The Average Precision Histograms presented on the TREC evaluation result pages are an attempt to show what is happening at the individual topic level.

This large topic/query variability presents a great opportunity for improving system performance. If we can understand why some systems do well on some queries but poorly on others, then we can start introducing query dependent processing to improve results on those poor performance queries.

Unfortunately, we just don’t have enough information from the results of a typical TREC task to really understand what is happening. The results on 50 to 150 queries are just not enough to draw any conclusions. The Query Track at TREC is an attempt to gather enough information from a large number of systems on a large number of queries to be able to start understanding query variability.

1.1 Query vs Topic

For the purposes of this track, a *topic* is considered an information need of a user. It includes a full statement of what information is wanted as well as information the user knows that pertains to the request. A *query* is what the user actually types to a retrieval system. It is much shorter than a topic, but is the only information from the user that the system has. Topic 51 (the first topic used in the Query Track) is given below. A query corresponding to Topic 51 might be something as simple as “Airbus subsidies”.

TOPIC 51

<top>

<head> Tipster Topic Description

<num> Number: 051

<dom> Domain: International Economics

<title> Topic: Airbus Subsidies

<desc> Description: Document will discuss government assistance to Airbus Industrie, or mention a trade dispute between Airbus and a U.S. aircraft producer over the issue of subsidies.

<smry> Summary: Document will discuss government assistance to Airbus Industrie, or mention a trade dispute between Airbus and a U.S. aircraft producer over the issue of subsidies.

<narr> Narrative: A relevant document will cite or discuss assistance to Airbus Industrie by the French, German, British or Spanish government(s), or will discuss a trade dispute between Airbus or the European governments and a U.S. aircraft producer, most likely Boeing Co. or McDonnell Douglas Corp., or the U.S. government, over federal subsidies to Airbus.

<con> Concept(s):

1. Airbus Industrie

2. European aircraft consortium, Messerschmitt-Boelkow-Blohm GmbH, British Aerospace PLC, Aerospatiale, Construcciones Aeronauticas S.A.

3. federal subsidies, government assistance, aid, loan, financing

4. trade dispute, trade controversy, trade tension

5. General Agreement on Tariffs and Trade (GATT) aircraft code

6. Trade Policy Review Group (TPRG)

7. complaint, objection

8. retaliation, anti-dumping duty petition, countervailing duty petition, sanctions

<def> Definition(s): ...

1.2 Issues to Examine

There are a number of issues that we wish to examine in this and future Query Track experiments. They include

- Can we distinguish between easy and hard queries/topics?
 - Are queries hard or are topics hard?
 - Even if we can distinguish this from the results, can NLP analysis of a query distinguish this before-hand?
- What categories of queries can potentially yield performance differences?
- Where do query performance differences come from?
 - Examine system vs topic vs query.
- Can we easily create test collections with large numbers of queries with judgments?

If we can answer these questions, then we may make it possible to improve retrieval systems dramatically.

2 Query Track Test Collection Creation

The construction of the Query Track test collection consists of 2 sub-tasks. In the first sub-task, groups take each of topics 51-100 from TREC 1 and create one or more queries based on the topic. In the second sub-task, each group runs one or more versions of their

system on all the queries from all the groups. The results are then evaluated and analysis can begin!

2.1 Query Creation Sub-Task

Groups create one or more versions of each of TREC topics 51-100 in categories

- Very short: 2-4 words based on the topic and possibly a few relevant documents from TREC disk 2.
- Sentence: 1-2 sentences using topic and relevant documents.
- Sentence-Feedback only: 1-2 sentences using only the relevant documents. The aim is to increase vocabulary variability.
- Weighted terms: lists of unstemmed terms with weights, possibly obtained through feedback on relevant documents from TREC disk 2.

The five participating groups produced 23 Query Sets. Each query set consisted of 50 queries corresponding to topics 51-100, for a total of 1150 queries. 15 Query Sets were produced by students and the rest by experts (retrieval system designers).

APL	INQ	Sab	Acs	Pir
Johns Hopkins	Umass	Sabir	Acsys	Queens
Expert	Students	Expert	Expert	Expert
2 weighted terms	5 short 5 sentence 5 feedback	3 short 1 feedback	1 short	1 short

Several versions of queries for topic 51 are given below. It was quite surprising how few duplicate queries there were, about 16%.

Sample of queries for Topic 51

- 51 01 recent airbus issues
- 51 02 Airbus subsidies dispute
- 51 03 Airbus subsidy battle
- 51 04 Airbus subsidies dispute
- 51 05 U.S. Airbus subsidies
- 51 06 What are the reactions of American companies to the trade dispute and how the dispute progresses?
- 51 07 What are the issues being debated regarding complaints against Airbus Industrie?
- 51 08 News related to the Airbus subsidy battle.
- 51 09 U.S. and Europe dispute over Airbus subsidies
- 51 10 Is European government risking trade conflicts over issue of Airbus subsidies?
- 51 11 How is the Airbus business in the world ?
- 51 12 why did the US put duties on airbus?

2.2 Retrieval Sub-Task

After the Query Sets were constructed, they were distributed to all the groups to run one or more retrieval runs on the TREC Disk 1 document collection (about 510,000 documents). The five groups performed 9 retrieval runs:

- APL : 1 run - words plus blind feedback
- INQ: 3 runs
 - only query terms
 - query terms plus structure
 - query terms plus structure plus blind feedback
- Sab: 3 runs
 - query terms plus adjacency phrases
 - query terms plus phrases plus 6 terms expansion from blind feedback
 - query terms plus phrases plus 27 terms expansion
- acs: 1 run - no expansion, base run
- pir: 1 run - blind feedback

The groups submitted the results (top 1000 documents retrieved for each query) to NIST for evaluation. There were a total of 203 runs; not all groups were able to run the 2 weighted term query sets. Thus the total was 9 runs * 21 NL queries plus 7 runs * 2 weighted terms queries.

The runs evaluated at NIST using trec_eval, concentrating on Mean Average Precision. The results of the initial evaluation were given to the five groups. This included

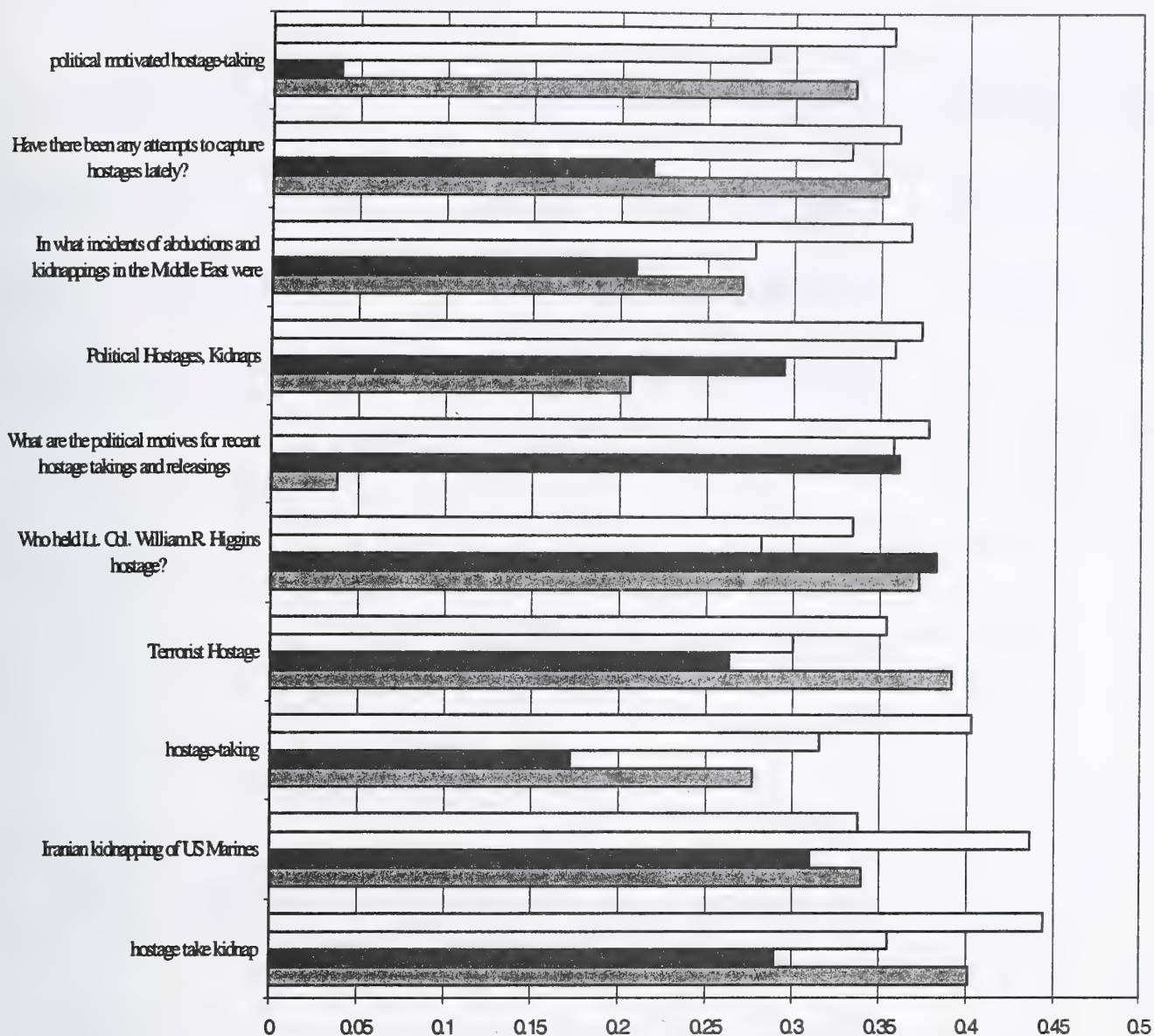
- Rankings of all documents (440 Mbytes in size)
- MAPs of all groups on all queries
- Various averages and standard deviations

3 Query Track Analysis

We present a very preliminary analysis of some aspects of the Query Track data. Other groups, notably the APL group of Johns Hopkins, have done more analysis. In addition, Walter Liggett of NIST has a paper in this proceedings.

3.1 Individual Query Analysis

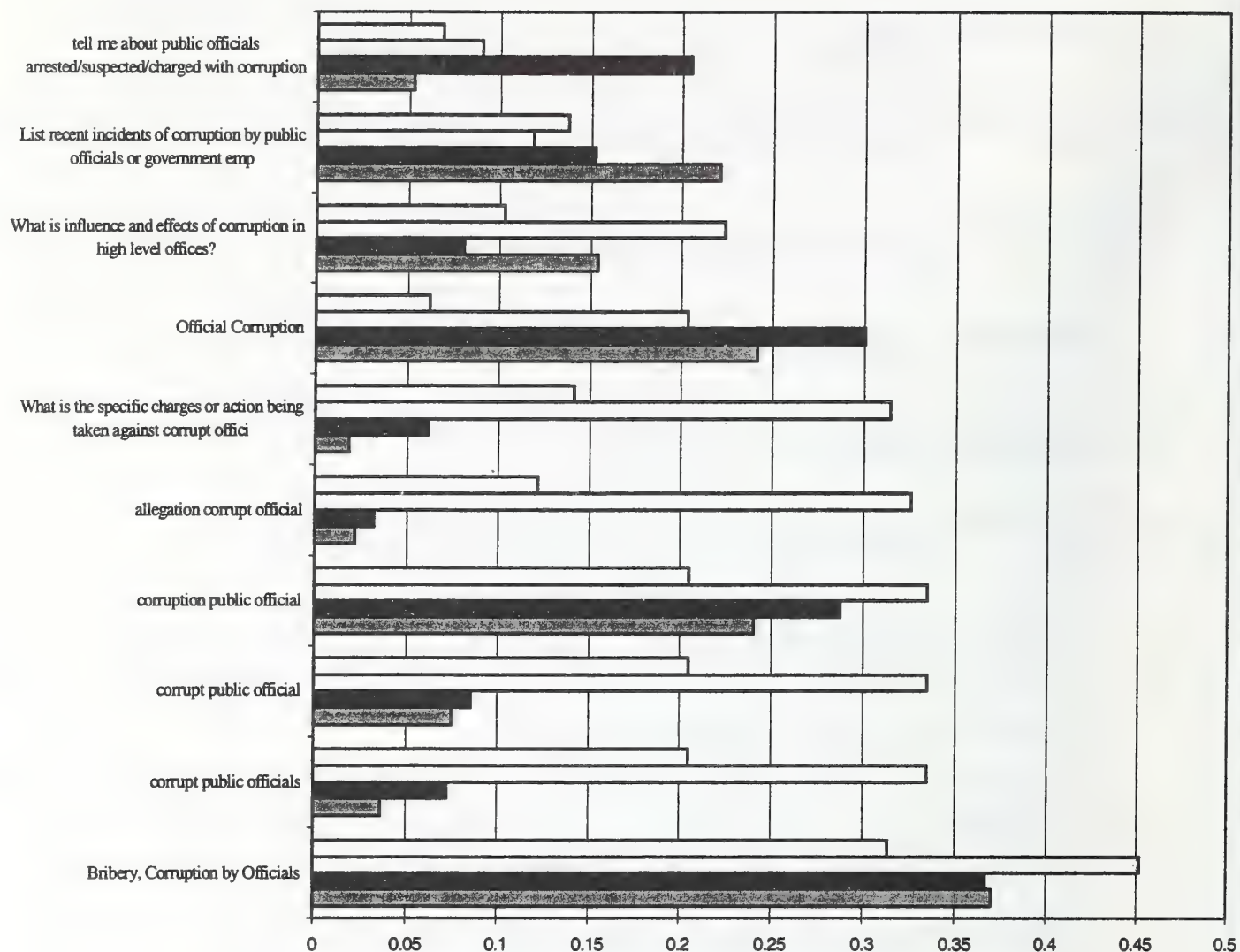
We look at the performance of 4 good runs on the top 10 queries per topic. The PIR, INQe, Sabe, and APL runs are the best runs of their respective groups, all using their own version of query expansion based on blind feedback. We want to examine how performance varies due to both system differences and query differences. Here, we look at how the 4 systems do on 4 topics, looking qualitatively at outliers, and doing an analysis of variants on each query.



ANOVA

Source of Variance	SS	df	MS	F	P-value	F crit
Rows	0.049356	9	0.005484	0.820275	0.602844	2.250133
Columns	0.07319	3	0.024397	3.649156	0.024951	2.960348
Error	0.180511	27	0.006686			
Total	0.303057	39				

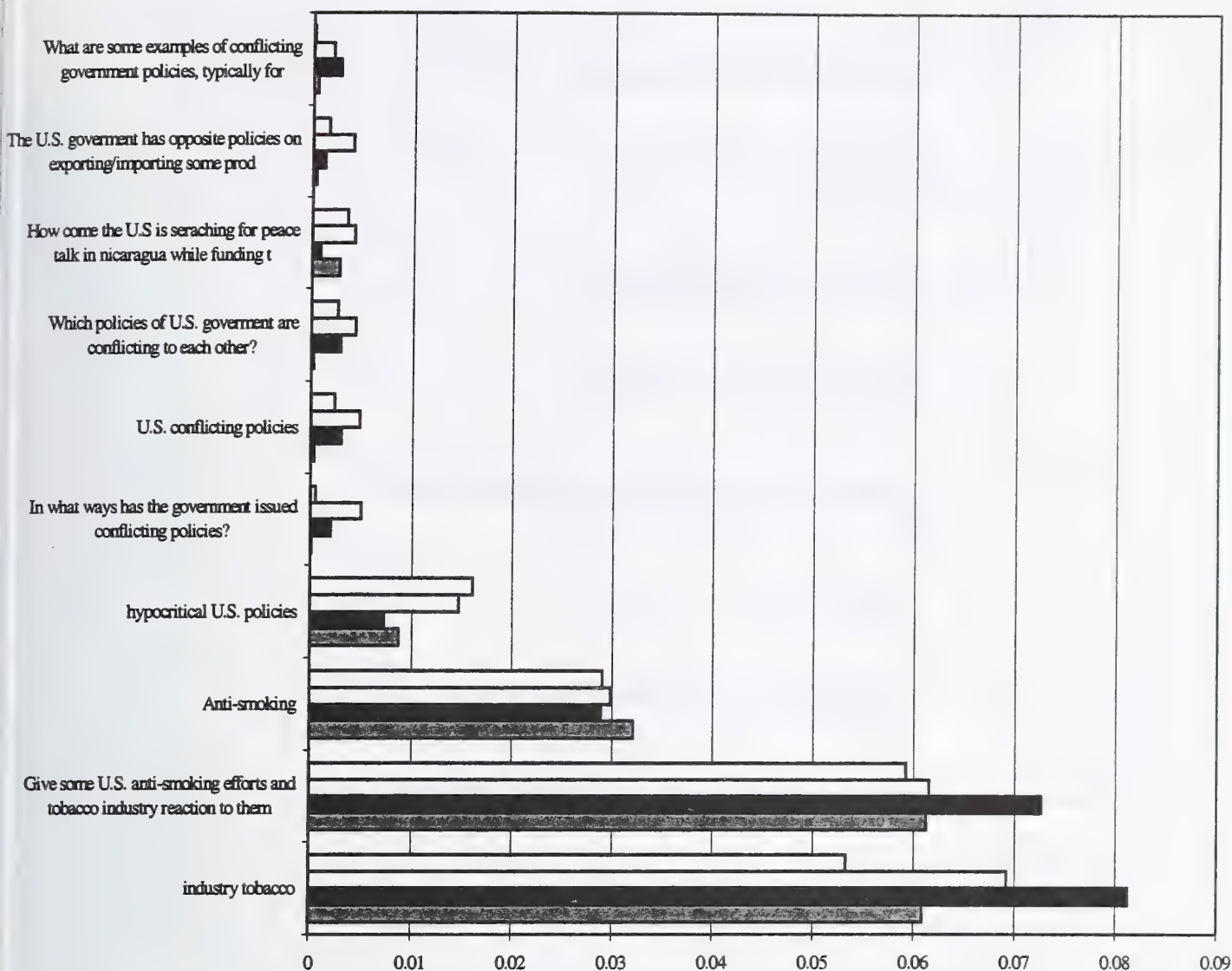
Topic 64



ANOVA

ce of Vari	SS	df	MS	F	P-value	F crit
Rows	0.2374	9	0.0264	3.8435	0.0031	2.2501
Columns	0.1078	3	0.0359	5.2333	0.0056	2.9603
Error	0.1853	27	0.0069			
Total	0.5305	39				

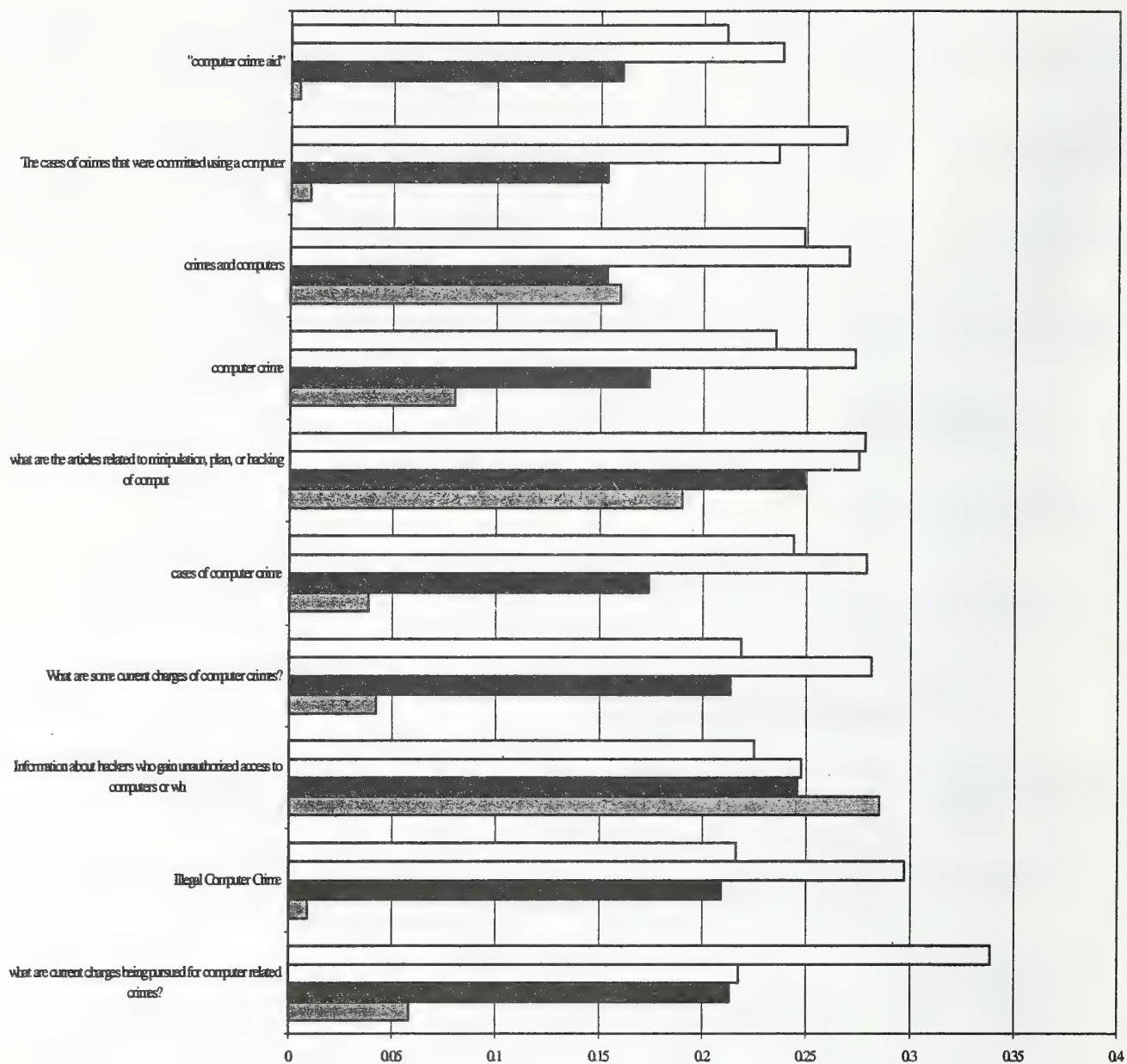
Topic 85



ANOVA

Source of Vari	SS	df	MS	F	P-value	F crit
Rows	0.0243	9	0.0027	134.5	5E-20	2.2501
Columns	0.0001	3	4E-05	1.8921	0.1548	2.9603
Error	0.0005	27	2E-05			
Total	0.025	39				

Topic 74



ANOVA

ce of Vari	SS	df	MS	F	P-value	F crit
Rows	0.0359	9	0.004	1.4476	0.2177	2.2501
Columns	0.1875	3	0.0625	22.681	2E-07	2.9603
Error	0.0744	27	0.0028			
Total	0.2978	39				

Topic 94

In Topic 64, all of the queries do well in general, but some of the systems do poorly for one or two queries. For example, the third system has problems with the hyphenated query “hostage-taking”, handling it inappropriately here. This sort of analysis highlights the system ‘blunders’ well; showing clearly that a system has a problem with a particular query syntax.

Topic 85 is more interesting. It is another easy topic, but one where there is a large variation due to both systems and queries. Some systems are doing better than others at focusing in on the key words in the longer queries; the second system does better with the shorter queries while the fourth system likes the longer queries. All the systems do well with a good short query that is augmented by a specific concept like “bribery”. Again, you can see the differences in the systems due to stemming and word order (phrases).

In Topic 74, the systems all behave the same (at a low level of performance), but the queries differ greatly. Performance improves as the queries shift from a general conceptual query, to a particular example. Obviously, this is a case where the topic itself is difficult.

Finally, in Topic 94 the systems are different, but the queries behave the same. The first three systems are all reasonably consistent across the queries, but the fourth system varies dramatically across queries.

In general, looking across all the topics, while using the 4 systems on the top 10 queries, we conclude that

- The queries provide a significant source of variance about half the topics.
- The top 4 systems are generally significantly different only due to “blundered runs”(e.g., stemming, hyphenation, spelling errors).

Looking at only the top 10 queries means we avoid the effect of “blundered queries”. Most topics have one or two queries that are simply inappropriate for the topic. For example, query 51-06 in the earlier list of queries for topic 51 is such a blundered query; it talks about *the dispute* without ever mentioning that the dispute is *airbus subsidies*. However, restricting analysis to the top 10 queries also means we avoid hard, but good, variants of the topics.

If we do an analysis of variance for each topic working with the entire set of results (all queries and all systems), we find that queries and systems almost always provided significant sources of variation, with the variation due to query generally much higher than the variation due to system. But it is impossible draw any conclusions from this given the presence of blundered queries, and the fact that we had multiple versions of the same basic system for SMART and INQUERY engines that are designed to be at different levels of effectiveness.

3.2 Query Type Analysis

The 21 natural language queries can be broken apart based upon the original category of their formation.

	Number of queries in set	Average MAP
Short Queries	10	.227
Long Sentences	6	.209
Long Feedback Sent	5	.146
Long (overall)	11	.183

The short queries do noticeably better than the longer queries, contrary to what would normally be expected. Analysis done by Walter Liggett elsewhere concluded that the long queries are much more variable: often a long query is the best query version for a topic, but more often a long query is also the worst query version. However, it is hard to say whether this is really a length factor or just a query origination factor. Half of the short queries were done by experts and half by students, but only 1 out of the 11 long query sets were done by an expert. This question needs to be re-examined when this confounding factor can be removed.

RunSet	MAP
APL	.216
INQa	.167
INQp	.194
INQe	.229
Saba	.205
Sabm	.224
Sabe	.244
Acs	.147
Pir	.224

The table above gives the performance of the 9 system variations averaged across all the queries. Note the performance increase among the INQUERY and SMART (Sabir) systems as query structure and query expansion terms are added. The differences between the different versions of the same overall system are significant. The differences between the top 4 systems (APL, INQe, Sabe, Pir) are not significant. Note that the scores are much higher (ranging from .288 to .329) when averaged only over the top 10 queries per topic. These scores are much closer to the original TREC 1 scores, where systems had access to the entire long topic statement.

4 Conclusion

We've reaffirmed the tremendous variation that sometimes gets hidden underneath the averages of a typical IR experiment.

- Topics are extremely variable
- Queries dealing with the same topic are extremely variable. Even short queries were rarely duplicated (16%).
- Systems were only somewhat variable.

The lack of system variability could be due to the particular systems involved. They are all “bag-of-words” statistical systems, with the good systems all doing either implicit or explicit blind feedback query expansion. We need to repeat this experiment with more systems of different types.

We examined differences between using long or short queries. In this experiment, the short queries performed better. That could be because the particular systems being tested were not set up to take into account the relationships between query words that full sentences give you. On the other hand, students constructed almost all the long queries while experts constructed half of the short queries, so we could just be seeing a user experience effect. This experiment needs to be repeated.

We have started to analyze components of variance. However, there were a limited number of independent systems being tested. It is clear we need many more systems before we can reach conclusions here.

More systems would also be useful for learning to distinguish between a poor query, and a good query that is hard. The current operational definition of a good hard query is a query on which one system does well, but other systems do poorly. This implies enough information exists in the query, but that the state of the art is such that most systems cannot take advantage of the information. A collection of good hard queries might be especially useful for developing future systems.

The query collection as it exists is already a major resource for future experiments.

- One of the only query collections with spelling and other mistakes!
- Excellent test-bed for system tuning. Comparisons within a topic are valuable: what query syntax does a system not handle well?
- Provides a large number of queries (1150) with relevance judgments. This will be quite useful as systems start to do NLP analysis of queries.
- Provides repeatable, but non-identical, experiments in a controlled environment.

This last point may be especially valuable because it enables experiments of a type we have not been able to do before. If we view a particular retrieval task as responding to a given information need with a set of good documents (the relevant documents for that topic), we now have 23 different ways to accomplish that task (actually, a few less than 23 because of query blunders and duplication). We can start to study variability of approaches; are some approaches more stable than others? Eliminating topic variability from such studies is essential.

Analysis of the Query Track data has just begun; there is a wealth of data available. We encourage you all to play with the data and to add to it in future Query Tracks. Who knows what we will find in the future!

The TREC-8 Question Answering Track Report

Ellen M. Voorhees
National Institute of Standards and Technology
Gaithersburg, MD 20899
ellen.voorhees@nist.gov

Abstract

The TREC-8 Question Answering track was the first large-scale evaluation of domain-independent question answering systems. This paper summarizes the results of the track by giving a brief overview of the different approaches taken to solve the problem. The most accurate systems found a correct response for more than 2/3 of the questions. Relatively simple bag-of-words approaches were adequate for finding answers when responses could be as long as a paragraph (250 bytes), but more sophisticated processing was necessary for more direct responses (50 bytes).

The TREC-8 Question Answering track was an initial effort to bring the benefits of large-scale evaluation to bear on a question answering (QA) task. The goal in the QA task is to retrieve small snippets of text that contain the actual answer to a question rather than the document lists traditionally returned by text retrieval systems. The assumption is that users would usually prefer to be given the answer rather than find the answer themselves in a document.

This paper summarizes the retrieval results of the track; a companion paper ("The TREC-8 Question Answering Track Evaluation") gives details about how the evaluation was implemented. By necessity, a track report can give only an overview of the different approaches used in the track. Readers are urged to consult the participants' papers elsewhere in the Proceedings for details regarding a particular approach.

1 The Task

A successful evaluation requires a task that is neither too easy nor too difficult for the current technology. If the task is too simple, all systems do very well and nothing is learned. Similarly, if the task is too difficult, all systems do very poorly and again nothing is learned. Accordingly, we chose a constrained version of the general question answering problem as the focus of the track.

The document collection used in the task was the same as the TREC-8 ad hoc collection, namely the set of documents on TREC disks 4 and 5 minus the *Congressional Record* documents. The documents consist mostly of newspaper articles and thus contain information on a wide variety of subjects. Participants were given 200 fact-based, short-answer questions, such as those given in Figure 1. Each question was guaranteed to have at least one document in the collection that explicitly answered the question.

Participants returned a ranked list of five [*document-id*, *answer-string*] pairs per question such that each answer string was believed to contain an answer to the question. Answer strings were limited to either 50 or 250 bytes, and could either be extracted from the corresponding document or automatically generated from information contained in the document. Human assessors read each string and made a binary decision as to whether the string actually did contain an answer to the question in the context provided by the document. Taking document context into account allowed a system that correctly derived a response from a document that was incorrect to be given full credit for its response.

Given a set of judgments for the strings, the score computed for a submission was mean reciprocal rank, defined as follows. An individual question received a score equal to the reciprocal of the rank at which the first correct response was returned, or 0 if none of the five responses contained a correct answer. The score for a submission was then the mean of the individual questions' reciprocal ranks. The reciprocal rank has several advantages as a scoring metric. It is closely related to the average precision measure used extensively in document retrieval. It is bounded between 0 and 1, inclusive, and averages well. A run is penalized for

- How many calories are there in a Big Mac?
- What two US biochemists won the Nobel Prize in medicine in 1992?
- Who was the first American in space?
- Who is the voice of Miss Piggy?
- Where is the Taj Mahal?
- What costume designer decided that Michael Jackson should only wear one glove?
- In what year did Joe DiMaggio compile his 56-game hitting streak?
- What language is commonly used in Bombay?
- How many Grand Slam titles did Bjorn Borg win?
- Who was the 16th President of the United States?

Figure 1: Example questions used in the question answering track.

AT&T Labs Research	MultText Project	U. of Iowa
CL Research	New Mexico State U.	U. of Maryland, College Park
Cymfony, Inc.	NTT DATA Corp.	U. of Massachusetts
GE/U. of Pennsylvania	National Taiwan U.	U. of Ottawa
IBM Research	Royal Melbourne Inst. Technology	U. of Sheffield
LIMSI-CNRS	Seoul National U.	Xerox Research Centre Europe
MITRE	Southern Methodist U.	

Figure 2: Participants in the Question Answering track.

not retrieving any correct answer for a question, but not unduly so. However, the measure also has some drawbacks. The score for an individual question can take on only six values (0, .2, .25, .33, .5, 1). Question answering systems are given no credit for retrieving multiple (different) correct answers. Also, since the track required at least one response for each question, a system could receive no credit for realizing it did not know the answer.

2 Retrieval Results

Twenty different organizations participated in the Question Answering track. The participants are listed in Figure 2. A total of 45 runs were submitted, 20 runs using the 50-byte limit and 25 runs using the 250-byte limit. Table 1 gives both the mean reciprocal rank and the number of questions for which no answer was found for each run. (Two submissions that contained errors are omitted from the table.) The scores are computed over the 198 questions that comprised the official test set. The table is split between the 50-byte and the 250-byte runs and is sorted by decreasing mean reciprocal rank within run type.

The number of questions for which no answer was found shows that the most accurate systems were able to find an answer for more than 2/3 of the questions. Furthermore, when the answer was found at all it was usually ranked first, as shown by the fact that the mean reciprocal rank is also close to 2/3 for these systems.

While the run with the highest mean reciprocal rank score was a 50-byte run, a direct comparison between 50- and 250-byte submissions from the same participant shows that the 50-byte task is more difficult. For every organization that submitted runs of both lengths, the 250-byte limit run had a higher mean reciprocal rank. This is not a surprising result—a system has a greater chance of including a correct response in a

Table 1: Mean reciprocal rank (MRR) and number of questions for which no correct response was found (# not found) for Question Answering track submissions.

Run Name	Participant	MRR	# not found
textract9908	Cymfony, Inc.	.660	54
SMUNLP1	Southern Methodist U.	.555	63
attqa50e	AT&T Research	.356	109
IBMDR995	IBM	.319	110
xeroxQA8sC	Xerox Research Centre Europe	.317	111
umdqa	U. of Maryland	.298	118
MTR99050	MITRE	.281	118
IBMVS995	IBM	.280	120
nttd8qs1	NTT Data Corp.	.273	121
attqa50p	AT&T Research	.261	121
nttd8qs2	NTT Data	.259	120
CRL50	New Mexico State U.	.220	130
INQ634	U. of Massachusetts	.191	140
CRDBASE050	GE/U. of Pennsylvania	.158	148
INQ638	U. of Massachusetts	.126	158
shefinq50	U. of Sheffield	.081	182
shefatt50	U. of Sheffield	.071	184
UIowaQA3	U. of Iowa	.018	188
UIowaQA4	U. of Iowa	.017	193

a) Runs with a 50-byte limit on the length of the response.

SMUNLP2	Southern Methodist U.	.646	44
attqa250p	AT&T Research	.545	63
GePenn	GE/U. of Pennsylvania	.510	72
attqa250e	AT&T Research	.483	78
uwmt9qa1	MultiText Project	.471	74
mds08q1	Royal Melbourne Inst. Tech	.453	77
xeroxQA8lC	Xerox Research Centre Europe	.453	83
nttd8ql1	NTT Data Corp.	.439	79
MTR99250	MITRE	.434	86
IBMDR992	IBM	.430	89
IBMVS992	IBM	.395	95
INQ635	U. of Massachusetts	.383	95
nttd8ql4	NTT Data Corp.	.371	93
LimsiLC	LIMSI-CNRS	.341	110
INQ639	U. of Massachusetts	.336	104
CRDBASE250	GE/U. of Pennsylvania	.319	111
clr99s	CL Research	.281	115
CRL250	New Mexico State University	.268	122
UIowaQA1	U. of Iowa	.267	117
Scai8QnA	Seoul National U.	.121	154
shefinq250	U. of Sheffield	.111	176
shefatt250	U. of Sheffield	.096	179
NTU99	National Taiwan U.	.087	173
UIowaQA2	U. of Iowa	.060	175

b) Runs with a 250-byte limit on the length of the response.

longer string—but it was not a guaranteed result. That is, longer strings that include a correct response were not always a correct response themselves. Response strings that contained multiple entities of the same semantic type as the answer and did not specifically indicate which of the entities was the answer were marked as incorrect. For example, for the question *What is the capital of Kosovo?* the 50-byte response of

0 miles northwest of Pristina, five demonstrators

was judged correct, while the 250-byte response of

protesters called for military intervention to end "the Albanian uprising."
</P> <P> At Vucitrn, 20 miles northwest of Pristina, five demonstrators were
reported injured, apparently in clashes with police. </P> <P> Violent clashes
were also repo

was judged incorrect since it is unclear from the response whether the capital is Vucitrn or Pristina.

The submissions from AT&T Research Labs demonstrate that existing passage-retrieval techniques can be successful for 250-byte runs, but are not suitable for 50-byte runs [18]. Their question answering system used a traditional vector-based retrieval system to select 50 documents and then scored each sentence within those documents by the number of question words in the surrounding context. For the passage-based runs (attqa50p and attqa250p), the highest scoring sentences were returned as the response. For their "entity-based" runs (attqa50e and attqa250e), high scoring sentences were further processed by a linguistic module. The passage-based method was very competitive for the 250-byte limit, but was not nearly as successful when restricted to just 50 bytes. NTT Data Corporation note similar effects in their runs [20]. These results suggest that the relatively simple bag-of-words approaches that are successfully used in text retrieval are not sufficient for extracting specific, fact-based answers.

3 Retrieval Strategies

Many participants used a variant of the following general strategy to the question answering problem. The system first attempted to classify a question according to the type of its answer as suggested by its question word. For example, a question that begins with "who" (*Who is the prime minister of Japan?*) implies a person or an organization is being sought, and a question beginning with "when" (*When did the Jurassic Period end?*) implies a time designation is needed. Next, the system retrieved a small portion of the document collection using standard text retrieval technology and the question as the query. The system performed a shallow parse of the returned documents to detect entities of the same type as the answer. If an entity of the required type was found sufficiently close to the question's words, the system returned that entity as the response. If no appropriate answer type was found, the system fell back to best-matching-passage techniques.

This approach works well provided the query types recognized by the system have broad enough coverage and the system can classify questions sufficiently accurately. Most systems could answer questions that began with "who" very accurately. However, questions that sought a person but did not actually begin with "who" (*Name the first private citizen to fly in space. What Nobel laureate was expelled from the Philippines before the conference on East Timor?*) were much more difficult. More difficult still were questions whose answers were not an entity of a specific type (*What is Head Start? Why did David Koresh ask the FBI for a word processor?*). Of course, pattern matching on expected answer types was not fool-proof even when "good" matches were found. One response to the question *Who was the first American in space?* was Jerry Brown, taken from a document that says

As for Wilson himself, he became a senator by defeating Jerry Brown, who has
been called the first American in space.

Broadly speaking, each of the following organizations used a variant of the general strategy: AT&T Labs Research [18], CL Research [10], Cymfony, Inc. [19], GE/University of Pennsylvania [13], LIMSI-CNRS [5], MITRE Corporation [2], New Mexico State University [15], NTT Data Corporation [20], Southern Methodist University [12], University of Maryland [14], University of Ottawa/NCR [11], University of Sheffield [8], and Xerox Research Centre Europe [7]. The approach used by IBM Research [16] was very similar in spirit to

this approach except they located entities at indexing time and used a bag-of-words scoring metric that incorporated the entities, thus providing efficient retrieval at question-answering time. The University of Iowa [4] classified questions by type and used their filtering system to learn features of answers. Seoul National University [17] performed an initial document retrieval run and then selected phrases from top-ranking documents by extracting the immediate neighborhood of the highest-weighted question word. Finally, the MultiText Project [3], National Taiwan University [9], Royal Melbourne Institute of Technology/CSIRO [6], and University of Massachusetts [1] used traditional passage retrieval techniques alone.

4 Conclusion

The Question Answering track was the first large-scale evaluation of domain-independent question answering systems. The questions used in the track were deliberately constrained to fact-based, short-answer questions to make the task amenable to evaluation. Systems generally classified a question according to the type of its answer, and then performed a shallow parse of likely documents to find objects of the entailed type. The most accurate systems were able to answer more than 2/3 of the questions correctly. Existing passage-retrieval techniques were adequate for finding answers when relatively long responses were permissible, but more sophisticated processing was needed to focus on the answer itself.

There will be another Question Answering track in TREC-9, which will be mostly the same as the TREC-8 track. One change in the track will be to have a test set of 500 questions rather than 200 questions, and to have many fewer of the questions be constructed from a target document.

References

- [1] James Allan, Jamie Callan, Fang-Fang Feng, and Daniella Malin. INQUERY and TREC-8. In Voorhees and Harman [21].
- [2] Eric Breck, John Burger, Lisa Ferro, David House, Marc Light, and Inderjeet Mani. A sys called Qanda. In Voorhees and Harman [21].
- [3] G.V. Cormack, C.L.A. Clarke, C.R. Palmer, and D.I.E. Kisman. Fast automatic passage ranking (MultiText experiments for TREC-8). In Voorhees and Harman [21].
- [4] David Eichmann and Padmini Srinivasan. Filters, webs and answers: The University of Iowa TREC-8 results. In Voorhees and Harman [21].
- [5] Olivier Ferret, Brigitte Grau, Gabriel Illouz, Christian Jacquemin, and Nicolas Masson. QALC—the question-answering program of the Language and Cognition group at LIMSI-CNRS. In Voorhees and Harman [21].
- [6] Michael Fuller, Marcin Kaszkiel, Sam Kimberly, Corinna Ng, Ross Wilkinson, Mingfang Wu, and Justin Zobel. The RMIT/CSIRO ad hoc, q&a, web, interactive, and speech experiments at TREC 8. In Voorhees and Harman [21].
- [7] David A. Hull. Xerox TREC-8 question answering track report. In Voorhees and Harman [21].
- [8] Keven Humphreys, Robert Gaizauskas, Mark Hepple, and Mark Sanderson. University of Sheffield TREC-8 Q & A system. In Voorhees and Harman [21].
- [9] Chuan-Jie Lin and Hsin-Hsi Chen. Description of preliminary results to TREC-8 QA task. In Voorhees and Harman [21].
- [10] Kenneth C. Litkowski. Question-answering using semantic relation triples. In Voorhees and Harman [21].
- [11] Joel Martin and Chris Lankester. Ask Me Tomorrow: The NRC and University of Ottawa question answering system. In Voorhees and Harman [21].

- [12] Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Richard Goodrum, Roxana Gîrju, and Vasile Rus. LASSO: A tool for surfing the answer net. In Voorhees and Harman [21].
- [13] Thomas S. Morton. Using coreference in question answering. In Voorhees and Harman [21].
- [14] Douglas W. Oard, Jianqiang Wang, Dekang Lin, and Ian Soboroff. TREC-8 experiments at Maryland: CLIR, QA and routing. In Voorhees and Harman [21].
- [15] Bill Ogden, Jim Cowie, Eugene Ludovik, Hugo Molina-Salgado, Sergei Nirenburg, Nigel Sharples, and Svetlana Sheremtyeva. CRL's TREC-8 systems: Cross-lingual IR and Q&A. In Voorhees and Harman [21].
- [16] John Prager, Dragomir Radev, Eric Brown, Anni Coden, and Valerie Samn. The use of predictive annotation for question answering in TREC8. In Voorhees and Harman [21].
- [17] Dong-Ho Shin, Yu-Hwan Kim, Sun Kim, Jae-Hong Eom, Hyung-Joo Shin, and Byoung-Tak Zhang. SCAI TREC-8 experiments. In Voorhees and Harman [21].
- [18] Amit Singhal, Steve Abney, Michiel Bacciani, Michael Collins, Donald Hindle, and Fernando Pereira. AT&T at TREC-8. In Voorhees and Harman [21].
- [19] Rohini Srihari and Wei Li. Information extraction supported question answering. In Voorhees and Harman [21].
- [20] Toru Takaki. NTT DATA: Overview of system approach at TREC-8 ad-hoc and question answering. In Voorhees and Harman [21].
- [21] E.M. Voorhees and D.K. Harman, editors. *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*. Electronic version available at <http://trec.nist.gov/pubs.html>, 2000.

The TREC-8 Question Answering Track Evaluation

Ellen M. Voorhees, Dawn M. Tice
National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

The TREC-8 Question Answering track was the first large-scale evaluation of systems that return answers, as opposed to lists of documents, in response to a question. As a first evaluation, it is important to examine the evaluation methodology itself to understand any limits on the conclusions that can be drawn from the evaluation and possibly to find ways to improve subsequent evaluations. This paper has two main goals: to describe in detail how the evaluation was implemented, and to examine the consequences of the methodology on the comparative performance of the systems participating in the evaluation. The examination uncovered no serious flaws in the methodology, supporting its continued use for question answering evaluation. Nonetheless, redefining the specific task to be performed so that it more closely matches an actual user task does appear warranted.

1 Introduction

The Text REtrieval Conference (TREC) is a series of workshops designed to advance the state-of-the-art in text retrieval by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies. Evaluating competing technologies on a common test set has had the desired effect of increasing text retrieval system effectiveness as demonstrated, for example, by the doubling of performance of the SMART system since the beginning of TREC [1]. However, users generally would prefer to receive *answers* in response to their questions, as opposed to the document lists traditionally returned by text retrieval systems. The TREC-8 Question Answering Track is an initial effort to bring the benefits of large-scale evaluation to bear on the question answering task.

Of course, in general “question answering” is a wide field ranging from simple yes/no answers for true-false questions to the presentation of complex results synthesized from multiple data sources. Designing an evaluation entailed defining a specific task to be performed, including how correct responses are recognized and results scored. The track coordinators, Amit Singhal and Tomek Strzalkowski, came to the TREC-7 conference with a proposed track definition, which was discussed and revised during a TREC-7 track planning workshop. The task was further refined through discussions by participants and other interested parties on the track mailing list. The track was advertised by posting to natural language processing related mailing lists with potential participants referred to the track web site at <http://www.research.att.com/~singhal/qa-track.html>.

The following specification of the track eventually emerged. Participants received a large collection of documents and 200 fact-based, short-answer questions such as “How many calories are there in a Big Mac?” Each question was guaranteed to have at least one document in the collection that answered the question. Participants were to return a ranked list of five strings per question such that each string was believed to contain an answer to the question. Depending on the run type, answer strings were limited to either 50 or 250 bytes. Human assessors read each string and made a binary decision as to whether or not the string actually did contain an answer to the question. Individual questions received a score equal to the reciprocal of the rank at which the first correct response was returned (or 0 if none of the five responses contained a correct answer). The score for a run was the mean of the individual questions’ reciprocal ranks.

The results of the evaluation are reported elsewhere in this proceedings in the Question Answering Track Overview paper. The current paper examines the evaluation methodology itself. That is, the paper explores the appropriateness of the evaluation design and the validity of the conclusions that can be drawn. The next section describes how the design was implemented: how the test questions were selected, the instructions

given to the human assessors, and an analysis of how the assessors perceived their task. One of the main conclusions of this analysis is that even for these highly constrained questions, answers depend on context and different assessors have legitimate differences of opinion as to whether a particular answer string is correct. In light of this finding, section 3 examines the effect of different judgments on the comparative performance of the answering systems. As is true with differences in relevance judgments for document retrieval, the differences in answer judgments do affect absolute scores, but relative scores across different runs remain stable. Using assessor judgments for the question answering task is thus a valid way to compare answering system quality.

2 Implementation of the TREC-8 QA Evaluation

The description of the QA task given above is complete as far as it goes, yet a number of additional decisions must be made to actually implement that description. Experience with TREC document retrieval tasks has demonstrated that seemingly minor details in the implementation of a task can occasionally have far-reaching effects on the evaluation results. As an example, the introduction of the three best content words as the "Title" field of TREC topic descriptions in TREC-6 has altered the nature of the TREC topic statements. In this section, therefore, we present a detailed description of how the QA task was implemented, including how the questions were selected, how the assessors were trained, and the implications of the mean reciprocal rank scoring metric. We also include a qualitative analysis of how the assessors perceived their task.

2.1 Creating the question set

The QA task as defined required a test set of 200 short-answer questions. Our goal was to have the test set represent a wide spectrum of subjects and question types while meeting this general specification. To accomplish the goal we collected a pool of 1,837 candidate questions from four different sources: TREC QA participants, the NIST TREC team, the NIST assessors, and question logs from the FAQFinder system. Our intention was that these different sources would provide different kinds of questions. The TREC participants have detailed knowledge about how their systems work and might have used that knowledge to select questions that would stress the technology. NIST team members created questions mostly to investigate how to teach assessors to create questions, but also have technical knowledge of question answering systems. The assessors have limited technical knowledge regarding question answering systems, and so represent a general user's point of view. Nonetheless, the assessors created their questions from the test document collection specifically for the track, and thus their questions do not represent natural information-seeking behavior. The questions taken from the FAQFinder logs, on the other hand, were submitted to the FAQFinder system by undergraduate students who were genuinely interested in the answers to the questions¹. Appendix A lists the set of questions in the final test set. The table at the end of the appendix gives the source of each question.

The FAQFinder logs contained 1500 questions. A subset of approximately 100 questions was selected by first eliminating entries that were not in the form of a question or that asked about subject matter deemed inappropriate for a government-sponsored evaluation, and then selecting those questions most likely to have an answer in the test document collection. Starting from the top of that list, Dawn Tice used NIST's PRISE search engine to look for documents containing answers to the current question, stopping when 24 answers had been found. Sadly, the FAQFinder question "Where did the Voorhees family ancestors immigrate from?" had to be eliminated at this step.

The other three sources supplied answer strings and document ids with the candidate questions. Seven NIST assessors created 70 candidate questions (10 questions each). The assessors used PRISE to search the QA document collection. Their methodology entailed thinking of a topic of interest, entering key search terms, and reading the text of the document to form a question. As a result, many of the assessors' questions are back-formulations of sentences from the texts. For instance, question 151, "Where did Dylan Thomas die?" was extracted from document FT934-10120 which reads, "DYLAN Thomas died in New York 40 years ago next Tuesday."

¹The FAQFinder question logs were given to NIST by Claire Cardie of Cornell University, with permission of Robin Burke, the creator of the FAQFinder system who is now at the University of California, Irvine.

Four members of the NIST TREC Team submitted a total of 25 candidate questions. One of the team members used web search engines to create his questions, which were then verified as having answers in the QA document collection. The others in the team searched the QA document database in much the same manner as the assessors.

Finally, 242 candidate questions were submitted by 23 groups that signed up to participate in the track (though in the end not all of these groups were able to submit runs). NIST does not know what methods participants used to create the questions, but the range of question types suggests a variety of methods was used.

The 337 candidate questions from sources other than the FAQFinder logs were then filtered by the NIST team to select the final test set. Our main goal for the first running of the track was to create a set of clean, straightforward questions and answers. We eliminated any question that a member of the team thought was ambiguous, that had a list of three or more items for an answer, that had an answer string greater than 50 bytes, or that were much too obscure or contrived (i.e., extreme examples of back-formulations). Since we were unsure how difficult (or easy) the task would be for existing systems, we tried to select questions with a range of difficulty, including some questions that we felt would challenge the systems. For example, while most questions that required compound answers were eliminated, we kept a few questions that had compound answers by rewording the questions to indicate that a compound answer was required. Thus question 16, "What two US biochemists won the Nobel Prize in medicine in 1992?" required the names of both biochemists.

Once the set of 200 questions was selected, NIST checked the document ids and answer strings submitted by the source to ensure the answer was really there. To do so, we entered the supplied answer texts as search strings in PRISE. We found some instances of incorrect document numbers or answer strings for the questions as submitted, but were eventually able to find correct [answer string, document] pairs for all 200 questions (or so we thought).

Despite the care we took to select questions with straightforward, obvious answers and to ensure that all questions had answers in the document collection, once assessing began it became clear that there is no such thing as a question with an obvious answer. Not only did most questions have more different answers than we anticipated, but the assessors determined that two of the 200 questions had no clear answer. Question 131, "Which Japanese car maker had its biggest percentage of sale in the domestic market?" was submitted by a participant who supplied the answer of "Toyota" with a document that states "Toyota had 42% of the domestic market." However, the assessors were unsure whether "domestic market" referred to Japan or the United States, and refused to accept 42% as the largest percentage without further proof. Question 184, "When was Queen Victoria born?" was a FAQFinder question. Document FT924-6257 contains "Queen Victoria (1837-1901)," so we assumed the answer was 1837. Unfortunately, a closer reading of the document makes it clear that 1837 was the beginning of her reign, not of her life. Due to these problems, we eliminated questions 131 and 184 from the evaluation results.

Part of the reason we did not anticipate the variety of different answers for these questions was with the way we checked the answer strings. By searching for the answer text as supplied by the source, we were immediately put in the same mindset as the question author. Had we started from scratch not knowing the answer, we probably would have found many of the other answers. Of course, this would have substantially increased the cost of what was already a labor-intensive question selection process. Furthermore, it would not have made the questions any more obvious, or less ambiguous, but simply would have made us aware of the complexities at an earlier stage.

Prior to the release of the test set of questions, NIST released a development set of 38 questions. These questions came from the same sources as the test set, except no FAQFinder questions were included in the development set (since we had not yet verified any FAQFinder questions at the time the set was released). The development set included all of the different types of questions as in the test set, but we made no attempt to keep the proportion of questions of a given type the same in the two sets. None of the development set questions was included in the test set.

2.2 Assessor training

The rationale for using human assessors to evaluate a task is to incorporate the perceptions of the end-users of the technology into the evaluation to the greatest extent possible. This argument suggests we should give assessors minimal or even no training so we receive their natural reactions. However, we must also be able to interpret the results of an evaluation, and this argues for ensuring that different assessors have the same basic understanding of what their task is. Our experience with the document relevance judging task has demonstrated the importance of adequate training for the assessors [2]. Accordingly, the assessors who performed the QA task received special training developed specifically for the QA task. The purpose of the training was not to drill the assessors on a specific set of assessment rules, but rather to motivate their task and provide guidance on the sorts of issues that might arise while they were assessing the test questions.

To minimize any confusion between tasks, the assessors were required to finish their assessments for the TREC ad hoc task before being trained on the QA task. The assessors generally finished the ad hoc task at different times, so most QA training was on a one-on-one basis. At most, we had two assessors being trained simultaneously on the QA task. Fifteen different assessors were trained on the task.

2.2.1 The QA assessment system

Since the QA task has different requirements from document relevance judging, a new assessment system was created especially for the QA task, though the new assessment system was based on the relevance judging system the assessors are very familiar with. Given a question number, the QA assessment system displays the text of the question and each answer string to be judged. The system also displays the document id associated with each answer string,² with answer strings sorted by document id so all strings associated with the same document are adjacent to one another. To judge a string, assessors click on either the “yes” or “no” radio button located next to the answer string. Clicking on either the document id or the answer string displays the corresponding document in a separate window. To assist them in targeting the relevant areas of a document, assessors can enter a search string that is then highlighted in the document. The search used is an exact (case-insensitive) string match on the entire string. This is different from the document assessing system in which the assessors enter a set of search terms and all words that conflate to the same stem as any of the search terms are highlighted in the text. When we developed the QA assessment system we thought the exact string match was better-suited to the QA task than was the set of search terms. However, the assessors found this difference between the two assessment systems difficult, and generally searched using only one word at a time in the QA task.

The QA training session took approximately two hours, and consisted of a general introduction that motivated the task, system-based training from a visual training manual of the QA assessment system, and task-based training using four sample questions with small answer pools concocted by the NIST TREC team. As a first step, the assessor was asked to read the written instructions reproduced in Appendix B. Next, we taught the mechanics of the QA assessment system by having the assessor follow the steps in the training manual. The visual training manual consists of screen shots of the system annotated with explanations of the system’s features.

2.2.2 QA task training

The answer pools for the four sample questions used in the task-based training were concocted by the NIST TREC team to illustrate the types of issues the assessors would face when judging actual test questions. Since we did not have previous experience with this task, we were not certain exactly what issues would arise, but made our best guess from the discussions on the track mailing list and the categorization of questions we developed while selecting the final test set of questions. In the end, all of the issues included in the training, plus more, actually occurred in judging the test set questions.

The four training questions (described in detail below) were ordered roughly by difficulty of judging. First the assessor was given the following scenario to use for judging answer strings:

²Participants were required to submit a document id with each answer string.

Assume there is a user who trusts the answering system completely, and therefore does not require that the system provide justification in its answer strings. Your job is to take each answer string in turn and judge if this answer string alone were returned to the trustful user, would the user be able to get the correct answer to the question from the string.

The assessor then began judging the answer strings for the first training question. Assessors were asked to hold their questions until they had judged all the answer strings for a given training question. Once they finished the first question, we reviewed the judgments with them. We paid particular attention to the reasons for their judgments as we discussed the training questions.

The first training question was “Who was Johnny Mathis’s track coach?” and the correct answer is Lou Vasquez. This question is relatively straightforward, and we used it to introduce the fundamentals of QA judging to the assessors: that the answer strings would contain snippets of text that were not necessarily grammatically correct and might even contain word fragments; that the answer string did not need to contain justification to be counted as correct; that the assessors were to judge the *string* not the document from which the string was drawn (after eight years of judging documents, this lesson was sometimes a hard one to learn); that the document context must be taken into account; and that the answer string had to be responsive to the question. The pool also illustrated a problem specific to “who” questions, i.e., whether first name only, or last name only is sufficient for a correct response. In the case when only part of the name is given, we told the assessors that they should use their own judgment, though we did suggest that first name only was probably insufficient while last name only was probably sufficient. But document context then becomes an issue. We judged as incorrect answer strings that contained “Vasquez” when the document associated with the response was about Lupe Vasquez or Ruben Vasquez (individuals who are completely unrelated to Lou Vasquez). Another of the answer strings in this pool contained a list of names extracted from the document that contained the correct answer, “Lou Vasquez, O.J. Simpson, Ollie Matson and Johnny Mathis.” This string was also judged as *incorrect* despite the fact that the correct answer is contained within it. The reasoning behind judging this string as incorrect is that the user, given just this answer string, still does not know who the track coach is, though admittedly the field is narrowed significantly. This is the sort of “interference” referred to in the written instructions to the assessors and what was meant by insisting that the answer string be responsive to the question. If answer strings contained multiple entities that were of the same semantic category as the correct answer, but did not indicate which of those entities was the actual answer, the response was judged as incorrect.

The second training question, “Who is the President of the United States?” demonstrated a question for which the correct answer changes over time, thus making document context vital. Responses to questions phrased in the present tense were judged as correct or incorrect based on the time of the document associated with the response. In the sample answer pool we had strings containing the names of former Presidents extracted from a document that stated Clinton was President; these responses were incorrect. We also had strings where the same Presidents were named but the strings were associated with documents when they were President; these responses were correct. Another answer string was “Bush” taken from a document written when George Bush was President but dealing exclusively with shrubs. All the assessors judged that string as incorrect (and we agreed). We also used the string “Bush” associated with a document that was written on the eve of Bush’s inauguration. Since Bush was not yet officially President at the time of the document, we judged this response as incorrect. When we inserted the inauguration document into the sample answer pool we assumed we were getting overly convoluted in our examples. But an equivalent issue did arise in the actual pools. The pool for question 147, “Who is the Prime Minister of Japan?” contained many responses associated with documents that announced the resignation of Prime Minister Hosokawa.

The third training question, “What is the world’s population?” illustrated questions whose answers change over time and can be reported to different levels of accuracy or in different units. Once again, the document context was used to determine whether a particular figure was correct. Example answer strings of “world population of 5.5 billion,” “5.4bn,” and “5.7bn” were all judged as correct since the corresponding documents gave these figures for the current population of the world. However, the answer string that gave a figure for the world’s population extracted from a document that was discussing a historical number (the population at the time of WWII for a document written in the 1980s) was judged as incorrect. Similarly, predictions for the future population were also judged as incorrect. Assessors were warned to watch for unfortunate truncation in answer strings. For example, the answer string “.4bn” extracted from a document

that gives the world's population as 5.4bn is incorrect. There were similar issues that were not covered in the training material but arose during the actual assessing. Some systems removed punctuation from the document source before extracting answers, and occasionally suffered for it. "5 5 billion" was not an acceptable substitute for "5.5 billion." Money units disappeared this way, too. A response of "500" was not acceptable when the correct answer was "\$500." In general, answer strings that did not contain a unit of measure for questions that required a quantity as a response were incorrect.

The final training question was "How tall is the Statue of Liberty?" This question was one of the questions in the development set, and one of the track participants pointed out that there were a number of documents that talked about various replicas of the Statue of Liberty, each of which was a different height. NIST decided that a string that correctly extracted the height of a replica from a document that was clearly discussing a replica would be judged as *incorrect*, arguing once again that such an answer was not responsive to the question. Unless the question specifically stated otherwise, we assumed that any question regarding a famous entity was asking about *the* famous entity and not about imitations, copies, etc. Once again this issue was seen in the set of test questions. For test question 73, "Where is the Taj Mahal?" we accepted only Agra, India, not the Taj Mahal casino in Atlantic City, New Jersey, nor the Taj Mahal Hotel in Bombay. Similarly, questions 199 and 200 asked for the height of the Matterhorn (i.e., the Alp) and the replica of the Matterhorn at Disneyland, respectively. Correct responses for one of these questions were incorrect for the other.

2.3 Judging the test set

Our original intention was that each test question would be judged by one assessor. However, once the assessing started it became clear that the assessors could judge an entire question much faster than we had originally planned for, and that judging correctness was much less cut-and-dried than we had hoped for. As a result, we decided to have each question be judged independently by three assessors. This would allow us to build a high-quality judgment set and also to gain insight into the effect of human assessors on question answering evaluation.

On average, an assessor took approximately a half-hour to judge one question. To judge a question, the assessor needed to judge each answer string that was in that question's answer pool. The answer pool consisted of each distinct [doc id, answer string] pair in the set of 45 runs submitted to the QA track. The mean size of an answer pool was 191.6 pairs (minimum pool size was 169 pairs, maximum pool size was 207 pairs), and the pools contained a mean of 55.3 distinct documents (min 28, max 93). As expected, there was very little overlap in strings across runs.

In all, fifteen different assessors judged some QA question. Because the assessors started at different times (depending on when they finished their ad hoc assessing) and worked at different rates, different assessors judged different numbers of questions and judged questions in different orders. The only invariants in assigning questions to assessors were that each question was judged three times and that no assessor judged a question more than once. The three sets of judgments for one question were mostly independent of each other, though there were occasional discussions among the assessors about particularly interesting assessing situations.

2.3.1 Assessors perception of their task

During the assessing process, the assessors interacted freely with the NIST TREC team members, asking for clarification of the assessment guidelines and verifying their application of the guidelines to particular cases. The interaction provided an informal mechanism for us to learn how the assessors perceived their task. We also instituted two more formal methods for gathering this information. Every time an assessor started a question, he or she was given a sheet of paper to record the canonical answers (i.e., the simplest form of the answers) to the question. We asked that they write any other comments they had about the question on the same sheet, and most sheets were returned with comments. The most detailed information came from a series of "think-aloud" observations of assessors judging an entire question. During a think-aloud session, the assessor was asked to think aloud as he or she considered each answer string in the answer pool. An observer (Tice) recorded the comments as the assessor judged the strings. Interruptions by the observer were kept to a minimum, although assessors were occasionally reminded to think aloud. Eight think-aloud

sessions were held, one each with five different assessors on five different questions plus all three assessors on a sixth question. To eliminate any “start-up” effects in the observations, each of the assessors judged at least three questions before being observed. The sessions were performed in a separate room with only the assessor and the observer present.

We had two goals for the information we gathered from the assessors. First, we hoped to gain a better understanding of how the assessors perceived the task and how they actually judged the answer strings (as opposed to how they were guided to judge the answer strings). Second, we wanted to discover if there were specific aspects of the task that could be improved for future evaluations.

Our observations suggest that the assessors understood their task and could do it. Several commented on how enjoyable they found it. Since this was the first time they had done this type of assessing, they were sometimes surprised (or amused or frustrated) by what the systems returned. For some questions, most of the answer strings were associated with the document that contained the answer, but none or few of the strings actually contained the answer. For example no one successfully extracted Marlon Brando as the actor who played the part of the Godfather in the movie “The Godfather” (question 77). Frequently strings would be truncated immediately before the answer. The answer pool for question 1, “Who is the author of the book, ‘The Iron Lady: A Biography of Margaret Thatcher?’” contained not only the string “The Iron Lady; A Biography of Margaret Thatcher by” but also responses of Ronald Reagan, Giroux, Deirdre Bair, Alfred A. Knopf, Lady Dorothy Neville, and Samuel Beckett. The systems’ lack of true understanding of the text sometimes led to amusing responses. One response to question 21, “Who was the first American in space?” was Jerry Brown, taken from document LA110190-0188 which says “As for Wilson himself, he became a senator by defeating Jerry Brown, who has been called the first American in space.” (The answer string was marked as incorrect.) A similar response was returned for question 196, “Who wrote ‘Hamlet’?”: “‘Hamlet,’ directed by Franco Zeffirelli and written by . . . well, you know.” (This response was also judged as incorrect.)

Because we told the assessors the NIST TREC team created the answer pools for the training questions, many assessors believed we had a hand in creating the responses to the test questions as well, and didn’t really seem to believe our claims of complete ignorance as to how the systems had created their responses. The assessors often picked apart the questions word for word looking for “tricks.” They were unhappy with question 93, “Who first circumnavigated the globe?” because their research outside of NIST showed that Magellan died before the trip was completed. We had them accept Magellan as the answer anyway. They also objected to questions 131 and 184, which we subsequently removed from the test set.

The assessors generally followed the assessing guidelines, though we did find some common patterns of mistakes. As alluded to earlier, some assessors needed reminding to judge an answer string based on what the string itself contained rather than what the associated document contained. For example, for question 146, “In what year did Ireland elect its first woman president?” one assessor was observed marking strings that contained no year as correct because the document contained the year. Another pattern was to mark strings as incorrect because they did not contain supporting evidence for the correctness of the answer. This was not so much a problem when the answer string contained only the answer (“What is the capital of Kosovo?”—answer string “Pristina”) but when the answer string contained random other information (answer string “Arkan Calls For Expulsion of 700,000 Albanians AU0305195294 Pristina KOSOVA DAILY REPORT Nr. 347 in English 3 May 94 AU0 305195294 Pristina KOSOVA DAILY REPORT Nr. 347”). Of course, there were also just plain blunders: times when the assessor hit the wrong button or whatever. Frequently the assessors would catch the blunders and correct them, but inevitably there were some blunders that persisted.

There was one aspect of the judging task that caused the assessors significant difficulty—an aspect related to the way in which the QA task itself was defined. The track guidelines required participants to return a document with the answer string and allowed answer strings to be generated (i.e., the answer strings did not have to be extracted from the document returned). The document was returned to provide the context for judging the answer string. The context was used not only to provide a frame of reference for questions whose answer changes over time, but also to give credit to systems that correctly extract information from a document that is in error. There were a number of instances, however, when an answer string contained the correct answer but that answer could not possibly have been determined from the document returned. For example, the correct answer for question 193, “Who is the 16th President of the United States?” is

Abraham Lincoln. One of the answer strings returned contained Abraham Lincoln, but the associated document discussed Lincoln's Gettysburg Address. The document does not even mention that Lincoln was President, let alone that he was the 16th President. Because generated answers were allowed in the track, we instructed the assessors to judge these strings as correct. The assessors *hated* this, and found it very difficult to do. Giving credit for both a right answer when it was not supported by the document and a wrong answer when it was supported by the document is too weird and does not reflect a real user task. This became one of the largest sources of inconsistency among the assessors' judgments as some assessors stopped checking document context altogether and others failed to mark such an answer string as correct.

2.3.2 Differences among assessors

Many differences among assessors were not caused by mistakes, however, but by legitimate differences of opinion as to what constitutes an acceptable answer. Two prime examples of where such differences arise are the completeness of names and the granularity of dates and locations. For example, two assessors accepted "April 22" as a correct response to question 54, "When did Nixon die?" but the other assessor required the year as well. Year-only is almost always acceptable for historical questions, and even decade- or century-only is acceptable if the event in question is ancient enough. For question 160, "When did French revolutionaries storm the Bastille?", "July 14" and "1789" (as well as "July 14, 1789") were all considered acceptable for some assessors. Similar issues arise with locations. For question 191, "Where was Harry Truman born?" some assessors accepted only Lamar, Missouri, while others accepted just Missouri. No assessor accepted just USA, though for other questions country-only designations were judged as acceptable.

People are addressed in a variety of ways as well. The assessor training suggested that surname-only is usually acceptable while first-name-only seldom is. Besides obvious exceptions such as Cher or Madonna, there are the different forms of address in other cultures. For example, for question 40, "Who won the Nobel Peace Prize in 1991?" the full name of the recipient is Aung San Suu Kyi. Some assessors accepted all of "Aung San Suu Kyi," "Suu Kyi," "San Suu Kyi," and "Kyi."

These examples illustrate the myth of the obvious answer. It is pointless to try to create a set of rules that specify exactly what is acceptable or unacceptable in all cases, since the granularity of an acceptable response really does depend on the question and on the person receiving the answer. Even if it were possible to get assessors to judge exactly the same way all the time, that would defeat the purpose of the evaluation. Eventual end-users of the technology will have different opinions and expectations, and the technology will have to be able to accommodate those differences to be useful.

2.4 Scoring the results

With three sets of judgments for each question, we were able to form a high-quality judgment set as the final result of the assessment process. For each question, the three sets of assessor judgments were compared, and any [string, document] pair that had two different judgments was reviewed by an adjudicator (Voorhees). The adjudicator's role was not to provide a fourth judgment, but rather to decide if the differences in judgments were caused by differences of opinions or misapplication of the assessing guidelines. If a difference was a matter of opinion, the judgment of the majority of the assessors was used, even if the adjudicator would have judged it differently. If the difference was caused by an incorrect application of the judging guidelines, or caused the judgments to be inconsistent across the set of strings in the pool, the adjudicator overruled the majority opinion. Appendix C gives the total number of pairs in the pool, the number disagreed on, and the number of times the majority opinion was overruled by the adjudicator for each question.

On average, 6% of the answer strings that were judged were disagreed on, and 16% of the disagreements had the majority opinion overruled by the adjudicator. Looking at the total percentage of answer strings that had disagreements is somewhat misleading, though, since a large percentage of the answer strings are obviously wrong and assessors agree on those. Following the document relevance judgment literature [3], we can compute the *overlap* in the sets of strings that were judged correct. Overlap is defined as the size of the intersection of the sets of strings judged correct divided by the size of the union of the sets of strings judged correct. Thus, an overlap of 1.0 means perfect agreement and an overlap of 0.0 means the sets of strings judged as correct were disjoint. The mean overlap across all three judges for the 193 test questions that had at least 1 correct string found was .641. The table in Appendix C also gives the overlap for each question.

The QA track runs were scored using the adjudicated judgment set. Recall that a run consisted of a ranked list of up to five [answer string, document] pairs for each question, and that every pair from every run was judged. For each run, the scoring routine read the answer pairs for the current question in order. The current answer pair was located in the adjudicated judgment file. If the pair was judged as correct, the reciprocal of the current rank was computed and added to a running sum of reciprocal ranks for this run; the remaining responses for this question were ignored. If no correct pair was found for the question, the reciprocal was set to 0. After all questions were processed, the mean reciprocal rank was computed from the running sum, and both the mean and the number of questions for which no correct answer was found were written out.

The reciprocal rank has several advantages as a scoring metric. It is closely related to the average precision measure used extensively in document retrieval. It is bounded between 0 and 1, inclusive, and averages well. A run is penalized for not retrieving any correct answer for a question, but not unduly so. The measure also has some drawbacks that perhaps should be addressed in future evaluations. The score for an individual question can take on only six values (0, .2, .25, .33, .5, 1), so it is unlikely that parametric statistical significance tests would be appropriate for this task. Question answering systems are given no credit for retrieving multiple (different) correct answers. Also, since the track required at least one response for each question, systems could receive no credit for realizing they did not know the answer.

3 The Reliability of System Comparisons

One of the primary ways TREC has been successful in improving document retrieval performance is by creating appropriate test collections for researchers to use when developing their systems. Unfortunately, the TREC-8 QA track did not create a comparable QA test collection. The unit that was judged for correctness was the entire answer string. This is not comparable to judging documents in document retrieval test collections because different question answering runs almost never return exactly the same answer strings. Developing a true equivalent of document retrieval's relevance judgment sets for question answering is a high priority research problem, but for now it remains unsolved.

A second test collection issue is the reliability of the comparisons between judged runs. Since the preceding section makes it clear that assessor opinions do differ even for the simple questions with "obvious" answers that were used as test questions, there is little hope that more carefully defined assessor instructions or more proscribed question selection procedures will eliminate inconsistencies in judgments among assessors. We must therefore ensure that the relative effectiveness of two question answering strategies is insensitive to modest changes in the judgment set since no one judgment set represents a gold standard answer key. There is reason to believe that this may be the case. Relevance judgments for documents are also known to vary across different assessors [4], yet relative retrieval effectiveness is stable despite the differences [6]. This section investigates whether the stability of document retrieval system evaluation is also true for question answering system evaluation.

3.1 Defining different judgment sets

As described earlier, the judgment set used to score the TREC-8 question answering systems was created such that each individual question was judged by three different assessors. Any differences in the judgments among the three assessors were reviewed by an adjudicator who let the majority's judgment stand unless judgments were inconsistent across different answer strings or the assessing guidelines were not followed. This process reduces the number of blunders in the final judgment set and increases the likelihood that the stated assessing guidelines were actually followed, though it also more than triples the cost of creating a judgment set as compared to using a single assessor's judgments for each question. Since different judgments are available for each question, it is possible to directly measure the effect different judgments have on the systems' scores. Two questions need to be addressed: whether judgment sets that use a single assessor for a question ("one-judge qrels") are equivalent to one another, and if so, whether a one-judge qrels is an adequate substitute for the adjudicated qrels.

The procedure used to measure the effects of different judgment sets on final scores was identical to the one used to gauge the effect of differences in relevance judgments in document retrieval system evaluation [6],

namely quantifying the changes in *system rankings* when different qrels are used to score the runs. A system ranking is a list of the systems under consideration sorted by decreasing mean reciprocal rank. We used a correlation based on Kendall's tau [5] as the measure of association between two rankings. Kendall's tau computes the distance between two rankings as the minimum number of pairwise adjacent swaps to turn one ranking into the other. The distance is normalized by the number of items being ranked such that two identical rankings produce a correlation of 1.0, the correlation between a ranking and its perfect inverse is -1.0, and the expected correlation of two rankings chosen at random is 0.0.

With three judgments for each of 198 questions, we can form 3^{198} different one-judge qrels for this QA task. We generated a sample of 100,003 of these one-judge qrels³ by randomly selecting one of the assessors for each question and combining the selected judgments into one qrels. We then scored 41 of the QA runs using each of the 100,003 qrels (the remaining four runs were clear outliers and we removed them for this part of the investigation). We calculated the sample mean and standard deviation of the mean reciprocal rank for each run. The means are plotted in Figure 1 where the runs are sorted by decreasing mean. The error bars in Figure 1 indicate the minimum and the maximum mean reciprocal rank obtained for that run over the sample. Values for the means, standard deviations, and minimum and maximum scores qrels are given in the second column of Table 1.

In addition to the 100,003 one-judge qrels, we created four multiple-judge qrels. The first of these is the adjudicated qrels described above. The second is a straight majority opinion qrels set. This differs from the adjudicated set in that there was no overruling of the majority opinion when assessments differed. The remaining two qrels sets are the union and intersection sets. In the union qrels a response is considered to be correct if any assessor judged it correct; in the intersection qrels a response is considered to be correct if all three assessors judged it as correct.

The mean reciprocal rank scores for each of the runs for the four multiple-judge qrels are plotted along with the sample means in Figure 1, and are given in the first column of Table 1. These points demonstrate how the system ranking changes for a particular qrels versus the ranking by the mean: a run with a symbol higher than the corresponding symbol of a run to its left would be ranked differently in the particular qrels ranking. For example, the first two runs (SMUNLP2 and textract9908) would switch positions when evaluated by the adjudicated qrels set.

As is true for document retrieval evaluations, the absolute values of the scores do change when different qrels are used to evaluate the runs. The final column of Table 1 gives the number of questions whose score changes for that run depending on which individual assessor's judgments are used. However, we are interested in the effect on relative scores, which means we need to look at how the system rankings change when different qrels are used. We computed the mean of the Kendall correlations among the system rankings in two ways. In the first case, we took the mean of all pair-wise correlations in a random sample of 1000 of the one-judge rankings. In the second case, we took the mean of the Kendall's correlation between the adjudicated qrels and all 100,003 one-judge rankings. Finally we computed the correlation between the adjudicated ranking and each of the other multiple-judge rankings. The correlations are given in Table 2. The numbers in parentheses show the number of pairwise adjacent swaps a correlation represents given that there are 41 different runs being ranked. Since any two one-judge qrels are likely to contain exactly the same judgments for 1/3 of the questions on average, the qrels are not independent of one another. Thus the Kendall correlation shown may be slightly higher than it would be with completely independent qrels.

The correlations in the top part of Table 2 show that QA system rankings produced from one-judge qrels are at least as stable as document retrieval system rankings in the face of changes in judgments. There are minor differences in the rankings, but most of those differences are caused by runs whose mean reciprocal rank scores are very close. This answers our first question in the affirmative. One-judge rankings are essentially equivalent with one another for the purpose of comparative evaluation of QA systems.

The second half of Table 2 suggests that one-judge qrels are also equivalent to the expensive adjudicated qrels. As can be seen from Figure 1, the adjudicated score for a run always lies within the boundaries of the minimum and maximum scores obtained on the sample of one-judge qrels. This is not true for the union and intersection qrels, which is a difference between QA evaluation and document retrieval evaluation.

³In the document retrieval study, three of the qrels sets were special cases. They are not special cases for the QA task, but existing code computed them, so they were left in.

Table 1: Variation in mean reciprocal rank by judgment set ("qrels"). The runs are ordered by decreasing mean reciprocal rank using the official adjudicated qrels (Adj). The next three columns give the score obtained using the Majority (Maj), Union (Union), and Intersection (Inter) qrels. The next four columns give the distribution of the mean reciprocal rank over the set of 100,003 single-judge qrels sets: the sample mean (Mean), the sample standard deviation (σ), the minimum (Min), and the maximum (Max). The final column gives the number of questions whose score varies depending on which single judge assessments are used.

Run	Qrels				Over 100,003				# Qs
	Adj	Maj	Union	Inter	Mean	σ	Min	Max	
textract9908	.660	.622	.705	.518	.617	.013	.564	.676	48
SMUNLP2	.646	.648	.667	.566	.627	.010	.583	.662	28
SMUNLP1	.555	.530	.586	.395	.504	.014	.446	.557	49
attqa250p	.545	.549	.596	.485	.543	.010	.502	.582	32
GePenn	.510	.501	.541	.446	.496	.009	.458	.529	32
attqa250e	.483	.478	.528	.434	.480	.009	.439	.517	31
uwmt8qa1	.471	.468	.504	.410	.462	.008	.431	.492	37
mds08q1	.453	.452	.495	.405	.452	.008	.418	.486	33
xeroxQA81C	.453	.450	.493	.374	.440	.010	.396	.480	38
nttd8ql1	.439	.444	.497	.380	.441	.010	.398	.478	43
MTR99250	.434	.429	.484	.327	.415	.012	.367	.465	44
IBMDR992	.430	.438	.461	.388	.429	.008	.394	.459	21
IBMVS992	.395	.402	.435	.339	.393	.009	.355	.429	30
INQ635	.383	.380	.430	.297	.369	.010	.326	.408	46
nttd8ql4	.371	.364	.415	.276	.353	.011	.305	.402	45
attqa50e	.356	.360	.387	.315	.355	.008	.323	.384	25
LimsiLC	.341	.338	.384	.298	.340	.008	.308	.376	28
INQ639	.336	.328	.377	.281	.330	.009	.295	.365	31
CRDBASE250	.319	.319	.347	.265	.310	.008	.277	.343	28
IBMDR995	.319	.307	.345	.215	.288	.011	.233	.334	40
xeroxQA8sC	.317	.314	.363	.232	.303	.011	.257	.347	39
umdqa	.298	.291	.344	.239	.293	.009	.257	.330	35
clr99s	.281	.272	.322	.222	.273	.009	.238	.309	34
MTR99050	.281	.253	.319	.191	.257	.010	.217	.303	38
IBMVS995	.280	.283	.313	.212	.269	.010	.231	.306	31
nttd8qs1	.273	.264	.306	.200	.257	.009	.211	.293	36
CRL250	.268	.259	.292	.199	.250	.009	.211	.290	29
UIowaQA1	.267	.264	.300	.245	.270	.007	.246	.298	17
attqa50p	.261	.228	.277	.149	.218	.011	.174	.262	35
nttd8qs2	.259	.252	.283	.176	.238	.009	.197	.276	36
CRL50	.220	.205	.233	.147	.195	.008	.160	.226	29
INQ634	.191	.179	.220	.131	.178	.008	.145	.212	29
CRDBASE050	.158	.146	.195	.114	.152	.008	.122	.186	31
INQ638	.126	.124	.150	.094	.123	.006	.098	.146	22
Scai8QnA	.121	.119	.138	.087	.114	.006	.089	.137	20
shefinq250	.111	.106	.121	.071	.099	.007	.071	.121	10
shefatt250	.096	.086	.101	.076	.088	.005	.076	.101	5
NTU99	.087	.088	.101	.057	.083	.006	.060	.101	13
shefinq50	.081	.061	.086	.040	.063	.007	.040	.086	9
shefatt50	.071	.066	.076	.056	.066	.005	.056	.076	4
UIowaQA2	.060	.056	.082	.041	.059	.006	.041	.082	14

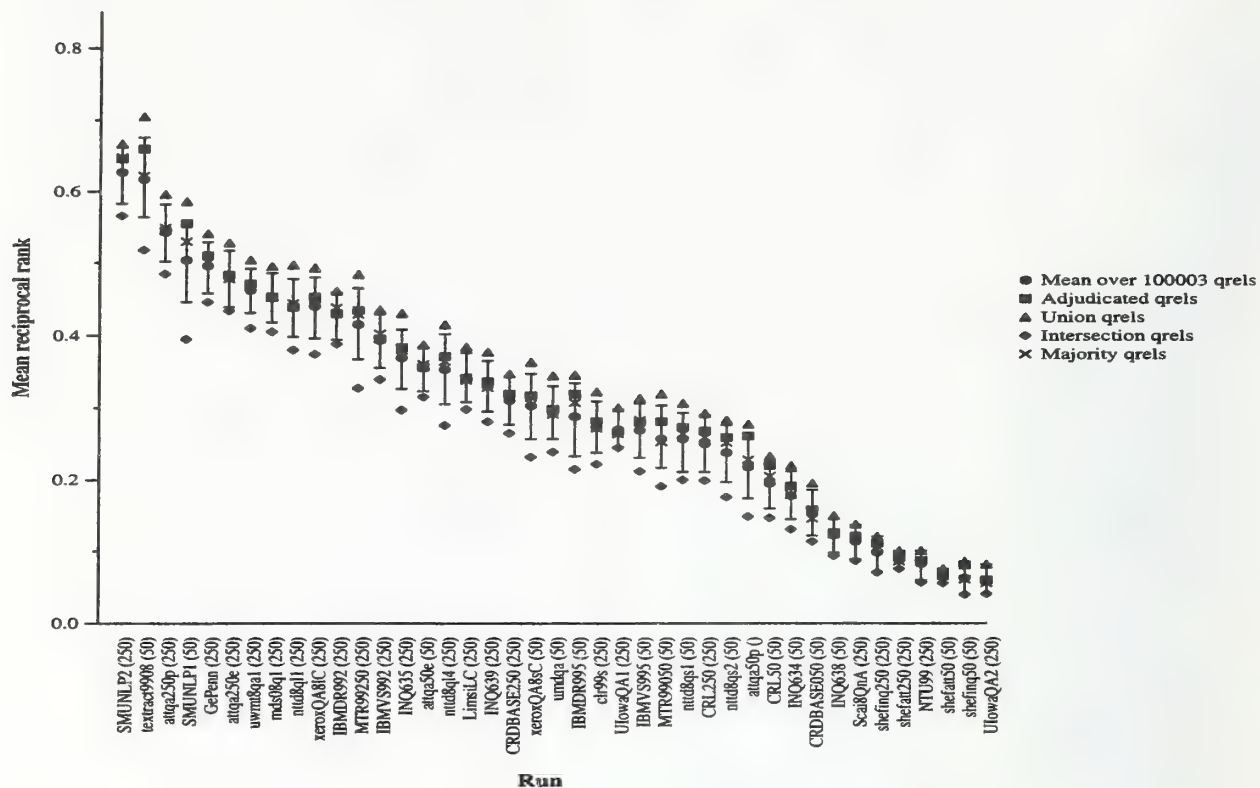


Figure 1: Sample mean, min, and max of the mean reciprocal rank computed for QA runs over a sample of 100,003 one-judge qrels. Also plotted are the mean reciprocal rank for the adjudicated, majority, union, and intersection qrels. Runs are labeled as either 50 byte limit (50) or 250 byte limit (250).

Table 2: Kendall correlation (τ) of system rankings and corresponding number of pairwise adjacent swaps produced by different qrels sets. With 41 systems, there is a maximum of 820 possible pairwise adjacent swaps.

	Mean τ	Min τ	Max τ
in subsample	.9632 (15.1)	.9171 (34)	.9976 (1)
with adjudicated	.9563 (17.9)	.9146 (35)	.9878 (5)

a) correlations for one-judge rankings

	τ
majority	.9683 (13)
union	.9780 (9)
intersection	.9146 (35)
a 1-judge qrels	.9683 (13)

b) correlations with the adjudicated ranking

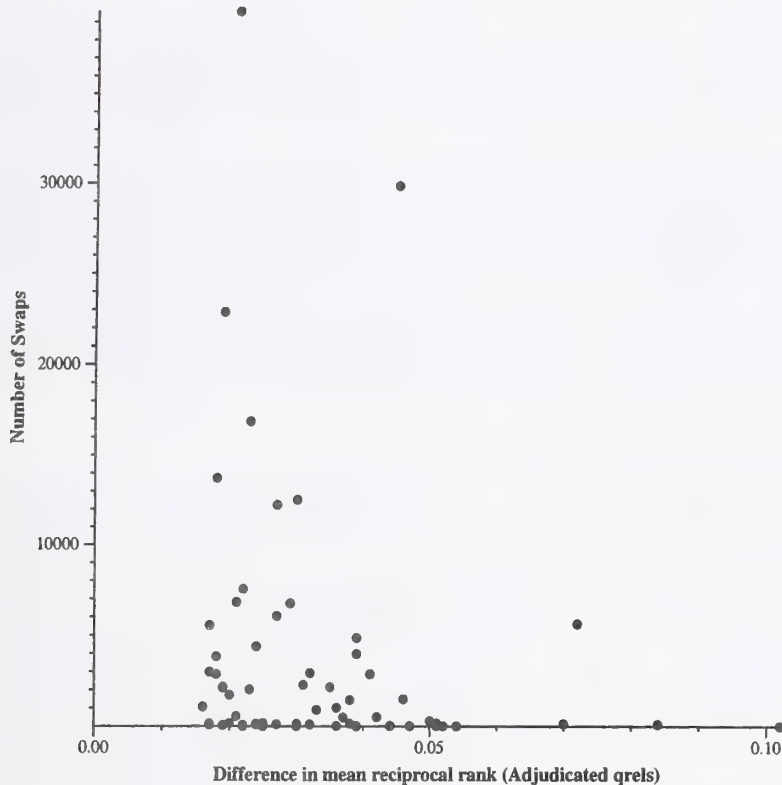


Figure 2: Number of swaps versus difference in mean reciprocal rank as computed using the Adjudicated qrels. Only pairs of systems that have a difference greater than .015 and that swapped more than 10 times are plotted.

3.2 Estimating the likelihood of a swap

While statements regarding the average stability of a set of runs are nice, researchers are often more interested in knowing the likelihood that any two particular runs will be ranked in a different order if the judgment sets change. Define a *swap* to be the situation in which one qrels ranks run i before run j and a second qrels ranks run j before rank i . We can use the sample of 100,003 one-judge qrels to count how often each pair of runs swaps.

Let $B[i, j]$ be the number of qrels that cause system i to evaluate as better than system j . Then the number of swaps for i and j is the smaller of $B[i, j]$ and $B[j, i]$ (assuming the larger number represents the “true” ranking). For the 820 pairs of runs under consideration, 123 had at least one swap. For pairs that had a difference greater than .015 in their mean reciprocal ranks according to the adjudicated qrels, only 85 ever swapped. As Table 1 shows, the standard deviation of the sample mean for a single run approaches .015 when qrels sets vary, so runs with smaller differences must be regarded as equivalent.

Figure 2 plots the number of swaps for a run pair against the difference in their mean reciprocal ranks according to the adjudicated qrels. Pairs with differences of less than .015 are not plotted, nor are pairs that swapped fewer than 10 times, leaving 66 pairs plotted. Points plotted furthest from the origin of the graph are of interest because they represent pairs of systems that either swap often or have large differences in the mean reciprocal rank. For example, the extreme point on the x-axis represents a difference of .102 in the mean reciprocal ranks between the SMUNLP1 run and the mds08q1 run. This pair of runs swapped only 14 times, however, thus having an estimated probability of a swap of only .0003 (14/50,000). At the other extreme, the umdqa and IBMDR995 runs swapped 39,567 times (estimated probability of .791) with a difference of .021 in the reciprocal ranks. While the difference in scores is (inversely) correlated with the probability that a pair of runs will swap, it is clearly not the only factor. Run pairs with much smaller differences between their scores swapped many fewer times than these two runs.

3.3 Question set size

It is important to remember that the system rankings used as a basis of this analysis were computed using the mean score over 198 questions. Using averages over a sufficient number of questions is vital to obtaining a stable evaluation.

What is sufficient? From a stability viewpoint, more questions in a test set is always better than fewer questions. But a test set with more questions is also more expensive to build than a set with fewer questions. With so little experience with the task, it is premature to set a final figure, though the TREC-8 evaluation does provide interesting data points. All of our analysis thus far has shown that 200 (or 198) is sufficient. To set a lower bound, note the last column in Table 1 that shows the number of questions whose score changed when the assessor changed for each run. Since some runs had almost 50 questions that were affected by judgment differences, a test set should probably have at least 100 questions.

4 Conclusion

The first running of any TREC track is more a test of the evaluation methodology used in the track than of the participating systems. This is particularly true with the TREC-8 QA track, which used an evaluation methodology based on human assessors for the question answering task. This paper validated the methodology used by showing it was both appropriate and effective.

The general question answering task was deliberately simplified for the TREC-8 track by constraining the questions to be fact-based, short-answer questions. The results of the track make it clear, however, that even for this highly-constrained version of the task legitimate differences of opinions exist as to whether a supplied answer string actually answers the question. Assessors differ on how much of a name is required, and on the granularity of times and locations. Having the evaluation accommodate differences of opinion in the answer keys reflects a requirement of the real problem; if assessors have different opinions on what constitutes an answer then eventual end-users of the technology will have different opinions as well. The technology must be able to accommodate user differences to be useful.

An evaluation is effective if the conclusions that can be drawn from it are meaningful and valid. The purpose of the TREC-8 evaluation was to compare different technologies for (a limited version of) the question answering task. The QA evaluation produced a nice spread of scores, and those runs that intuitively seemed better got higher scores. The comparisons are valid to the extent that they are stable under changes in the judgments that produce the scores. Indeed, our analysis suggests that the expensive adjudicated judgment set used in the track can be replaced with a single-opinion judgment, since the system rankings produced by single-opinion judgment sets are equivalent to one another and to the ranking produced by the adjudicated judgment set.

While on the whole the TREC-8 QA evaluation was sound, there was at least one problem with this implementation of the track. Since the track guidelines required a document be returned but allowed answer generation (as opposed to simple extraction), NIST made the attempt to judge as correct both strings that contained a wrong answer that had been correctly extracted from a mistaken document, and strings that contained a correct answer even though that answer could not be obtained from the information contained within the cited document. In retrospect, this was a bad decision. Taking document context into account only for certain situations was difficult for the assessors to do, and was one of the largest contributors to differences in judgments among the assessors. It also does not correspond well to any real user task. In future evaluations the task should either be a true question answering task wherein the system generates a response from any resource available to it and is evaluated strictly on whether the returned string contains a correct answer⁴ or an information extraction task in which the system extracts a response from text and is evaluated on successful extraction.

Another issue that must eventually be addressed is the fact that the current methodology does not create a true test collection. Currently judgments are based on entire answer strings because it is not yet clear how to map specific answer strings into more general answers. Unfortunately, this renders the judgments unsuitable for anything other than comparing the specific runs used to build the answer pools. The judgments do not constitute a generic judgment file for the QA task because the amount of overlap in answer strings

⁴Some mechanism whereby the assessors learn the set of correct answers is required.

across runs is quite small. The benefits of large-scale evaluation for question answering technology will not be fully realized until true test collections can be devised.

References

- [1] Chris Buckley, Mandar Mitra, Janet Walz, and Claire Cardie. SMART high precision: TREC 7. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 285–298, August 1999. NIST Special Publication 500-242. Electronic version available at <http://trec.nist.gov/pubs.html>.
- [2] Laura L. Downey and Dawn M. Tice. A usability case study using TREC and ZPRISE. *Information Processing and Management*, 35(5):589–603, 1999.
- [3] M.E. Lesk and G. Salton. Relevance assessments and retrieval system evaluation. *Information Storage and Retrieval*, 4:343–359, 1969.
- [4] Linda Schamber. Relevance and information behavior. *Annual Review of Information Science and Technology*, 29:3–48, 1994.
- [5] Alan Stuart. Kendall's tau. In Samuel Kotz and Norman L. Johnson, editors, *Encyclopedia of Statistical Sciences*, volume 4, pages 367–369. John Wiley & Sons, 1983.
- [6] Ellen M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. In W. Bruce Croft, Alistair Moffat, C.J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 315–323, Melbourne, Australia, August 1998. ACM Press, New York.

A The Test Set of 200 Questions

1. Who is the author of the book, "The Iron Lady: A Biography of Margaret Thatcher"?
2. What was the monetary value of the Nobel Peace Prize in 1989?
3. What does the Peugeot company manufacture?
4. How much did Mercury spend on advertising in 1993?
5. What is the name of the managing director of Apricot Computer?
6. Why did David Koresh ask the FBI for a word processor?
7. What debts did Qintex group leave?
8. What is the name of the rare neurological disease with symptoms such as: involuntary movements (tics), swearing, and incoherent vocalizations (grunts, shouts, etc.)?
9. How far is Yaroslavl from Moscow?
10. Name the designer of the shoe that spawned millions of plastic imitations, known as "jellies".
11. Who was President Cleveland's wife?
12. How much did Manchester United spend on players in 1993?
13. How much could you rent a Volkswagen bug for in 1966?
14. What country is the biggest producer of tungsten?
15. When was London's Docklands Light Railway constructed?
16. What two US biochemists won the Nobel Prize in medicine in 1992?
17. How long did the Charles Manson murder trial last?
18. Who was the first Taiwanese President?
19. Who was the leader of the Branch Davidian Cult confronted by the FBI in Waco, Texas in 1993?
20. Where is Inoco based?
21. Who was the first American in space?
22. When did the Jurassic Period end?
23. When did Spain and Korea start ambassadorial relations?
24. When did Nixon visit China?
25. Who was the lead actress in the movie "Sleepless in Seattle"?
26. What is the name of the "female" counterpart to El Nino, which results in cooling temperatures and very dry weather?
27. Where did the 6th annual meeting of Indonesia-Malaysia forest experts take place?
28. Who may be best known for breaking the color line in baseball?
29. What is the brightest star visible from Earth?
30. What are the Valdez Principles?
31. Where was Ulysses S. Grant born?
32. Who received the Will Rogers Award in 1989?
33. What is the largest city in Germany?
34. Where is the actress, Marion Davies, buried?
35. What is the name of the highest mountain in Africa?
36. In 1990, what day of the week did Christmas fall on?
37. What was the name of the US helicopter pilot shot down over North Korea?
38. Where was George Washington born?
39. Who was chosen to be the first black chairman of the military Joint Chiefs of Staff?
40. Who won the Nobel Peace Prize in 1991?
41. What is the legal blood alcohol limit for the state of California?

42. What was the target rate for M3 growth in 1992?
43. What costume designer decided that Michael Jackson should only wear one glove?
44. Who is the director of the international group called the Human Genome Organization (HUGO) that is trying to coordinate gene-mapping research worldwide?
45. When did Lucelly Garcia, a former ambassador of Columbia to Honduras, die?
46. Who is the mayor of Marbella?
47. What company is the largest Japanese ship builder?
48. Where is the massive North Korean nuclear complex located?
49. Who fired Maria Ybarra from her position in San Diego council?
50. When was Dubai's first concrete house built?
51. Who is the president of Stanford University?
52. Who invented the road traffic cone?
53. Who was the first doctor to successfully transplant a liver?
54. When did Nixon die?
55. Where is Microsoft's corporate headquarters located?
56. How many calories are there in a Big Mac?
57. What is the acronym for the rating system for air conditioner efficiency?
58. Name a film that has won the Golden Bear in the Berlin Film Festival?
59. Who was President of Costa Rica in 1994?
60. What is the fare cost for the round trip between New York and London on Concorde?
61. What brand of white rum is still made in Cuba?
62. What is the name of the chronic neurological autoimmune disease which attacks the protein sheath that surrounds nerve cells causing a gradual loss of movement in the body?
63. What nuclear-powered Russian submarine sank in the Norwegian Sea on April 7, 1989?
64. Who is the voice of Miss Piggy?
65. Name a country that is developing a magnetic levitation railway system?
66. Name the first private citizen to fly in space.
67. What is the longest river in the United States?
68. What does El Nino mean in spanish?
69. Who came up with the name, El Nino?
70. How many lives were lost in the China Airlines' crash in Nagoya, Japan?
71. In what year did Joe DiMaggio compile his 56-game hitting streak?
72. When did the original Howdy Doody show go off the air?
73. Where is the Taj Mahal?
74. Who leads the star ship Enterprise in Star Trek?
75. What cancer is commonly associated with AIDS?
76. In which year was New Zealand excluded from the ANZUS alliance?
77. Who played the part of the Godfather in the movie, "The Godfather"?
78. Which large U.S. city had the highest murder rate for 1988?
79. What did Shostakovich write for Rostropovich?
80. What is the name of the promising anticancer compound derived from the pacific yew tree?
81. How many inhabitants live in the town of Ushuaia?
82. How many consecutive baseball games did Lou Gehrig play?
83. What is the tallest building in Japan?

84. Which country is Australia's largest export market?
85. Which former Ku Klux Klan member won an elected office in the U.S.?
86. Who won two gold medals in skiing in the Olympic Games in Calgary?
87. Who followed Willy Brandt as chancellor of the Federal Republic of Germany?
88. What is Grenada's main commodity export?
89. At what age did Rossini stop writing opera?
90. Who is the founder of Scientology?
91. Which city in China has the largest number of foreign financial companies?
92. Who released the Internet worm in the late 1980s?
93. Who first circumnavigated the globe?
94. Who wrote the song, "Stardust"?
95. What country is the world's leading supplier of cannabis?
96. What time of day did Emperor Hirohito die?
97. How large is the Arctic refuge to preserve unique wildlife and wilderness value on Alaska's north coast?
98. Where is the highest point in Japan?
99. What is the term for the sum of all genetic material in a given organism?
100. What is considered the costliest disaster the insurance industry has ever faced?
101. How many people live in the Falklands?
102. Who is the Voyager project manager?
103. How many people died when the Estonia sank in 1994?
104. What language is most commonly used in Bombay?
105. How many people does Honda employ in the U.S.?
106. What is the second highest mountain peak in the world?
107. When was China's first nuclear test?
108. Which company created the Internet browser Mosaic?
109. Where does Buzz Aldrin want to build a permanent, manned space station?
110. Who killed Lee Harvey Oswald?
111. How long does it take to travel from Tokyo to Niigata?
112. Who is the President of Ghana?
113. What is the name of the medical condition in which a baby is born without a brain?
114. How much stronger is the new vitreous carbon material invented by the Tokyo Institute of Technology compared with the material made from cellulose?
115. What is Head Start?
116. Which team won the Super Bowl in 1968?
117. What two researchers discovered the double-helix structure of DNA in 1953?
118. What percentage of the world's plant and animal species can be found in the Amazon forests?
119. What Nobel laureate was expelled from the Philippines before the conference on East Timor?
120. Who held the endurance record for women pilots in 1929?
121. Who won the first general election for President held in Malawi in May 1994?
122. Who is section manager for guidance and control systems at JPL?
123. How many Vietnamese were there in the Soviet Union?
124. What was Agent Orange used for during the Vietnam War?
125. In what city is the US Declaration of Independence located?
126. When did Israel begin turning the Gaza Strip and Jericho over to the PLO?

127. Which city has the oldest relationship as a sister-city with Los Angeles?
128. Who was the second man to walk on the moon?
129. How many times was pitcher, Warren Spahn, a 20-game winner in his 21 major league seasons?
130. When was Yemen reunified?
131. Which Japanese car maker had its biggest percentage of sale in the domestic market?
132. What is the capital of Uruguay?
133. What is the name for the technique of growing certain plants in soils contaminated with toxic metals, wherein the plants take up the toxic metals, are harvested, and the metals recovered for recycling?
134. Where is it planned to berth the merchant ship, Lane Victory, which Merchant Marine veterans are converting into a floating museum?
135. What famous communist leader died in Mexico City?
136. Who is the Queen of Holland?
137. Who is the president of the Spanish government?
138. What is the name of the normal process in all living things, including humans, in which cells are programmed to "commit suicide"?
139. How many people did the United Nations commit to help restore order and distribute humanitarian relief in Somalia in September 1992?
140. How many people on the ground were killed from the bombing of Pan Am Flight 103 over Lockerbie, Scotland, December 21, 1988?
141. What is the duration of the trip from Bristol to London by rail?
142. What is the population of Ulan Bator, capital of Mongolia?
143. Where does most of the marijuana entering the United States come from?
144. How many megawatts will the power project in Indonesia, built by a consortium headed by Mission Energy of US, produce?
145. What did John Hinckley do to impress Jodie Foster?
146. In what year did Ireland elect its first woman president?
147. Who is the prime minister of Japan?
148. How many soldiers were involved in the last Panama invasion by the United States of America?
149. Where is the Bulls basketball team based?
150. What is the length of border between the Ukraine and Russia?
151. Where did Dylan Thomas die?
152. How many people live in Tokyo?
153. What is the capital of California?
154. How many Grand Slam titles did Bjorn Borg win?
155. Who was the Democratic nominee in the American presidential election?
156. When was General Manuel Noriega ousted as the leader of Panama and turned over to U.S. authorities?
157. Where is Dartmouth College?
158. How many mines can still be found in the Falklands after the war ended?
159. Why are electric cars less efficient in the north-east than in California?
160. When did French revolutionaries storm the Bastille?
161. How rich is Bill Gates?
162. What is the capital of Kosovo?
163. What state does Charles Robb represent?
164. Who is the leading competitor of Trans Union Company?
165. Which type of submarine was bought recently by South Korea?

166. When did communist control end in Hungary?
167. What nationality is Pope John Paul II?
168. Who was the captain of the tanker, Exxon Valdez, involved in the oil spill in Prince William Sound, Alaska, 1989?
169. Whom did the Chicago Bulls beat in the 1993 championship?
170. Who was President of Afghanistan in 1994?
171. Who is the director of intergovernmental affairs for the San Diego county?
172. Where is the Keck telescope?
173. How many moons does Jupiter have?
174. When did Jaco Pastorius die?
175. When did beethoven die?
176. How many people in Tucson?
177. How tall is Mt. Everest?
178. What is the capital of Congo?
179. What is the capital of Italy?
180. What is the capital of Sri Lanka?
181. What novel inspired the movie BladeRunner?
182. What was the first Gilbert and Sullivan opera?
183. What was the name of the computer in "2001: A Space Odyssey"?
184. When was Queen Victoria born?
185. When was the battle of the Somme fought?
186. Where did the Battle of the Bulge take place?
187. Where was Lincoln assassinated?
188. When was the women's suffrage amendment ratified?
189. Where is Qatar?
190. Where is South Bend?
191. Where was Harry Truman born?
192. Who was Secretary of State during the Nixon administration?
193. Who was the 16th President of the United States?
194. Who wrote "The Pines of Rome"?
195. Who wrote "Dubliners"?
196. Who wrote "Hamlet"?
197. What did Richard Feynman say upon hearing he would receive the Nobel Prize in Physics?
198. How did Socrates die?
199. How tall is the Matterhorn?
200. How tall is the replica of the Matterhorn at Disneyland?

Table 3: Sources of the 198 final test set questions. Two questions (131 and 184) were dropped from the evaluation after assessors determined there was no answer in the document collection. "Qid" is the question identifier. The remaining columns indicate the source of the corresponding question: "FAQ" for questions taken from the logs of the FAQFinder system, "Part" for questions submitted by TREC participants or the NIST TREC team, and "Assess" for questions submitted by the NIST assessors.

Qid	FAQ	Part	Assess	Qid	FAQ	Part	Assess	Qid	FAQ	Part	Assess	Qid	FAQ	Part	Assess
1			X	51		X		101		X		152		X	
2			X	52		X		102		X		153		X	
3		X		53			X	103		X		154		X	
4		X		54		X		104		X		155		X	
5		X		55		X		105		X		156			X
6		X		56		X		106		X		157		X	
7		X		57			X	107		X		158		X	
8			X	58		X		108		X		159			X
9		X		59			X	109		X		160		X	
10			X	60		X		110		X		161		X	
11		X		61		X		111		X		162		X	
12		X		62			X	112			X	163		X	
13		X		63			X	113			X	164		X	
14			X	64		X		114		X		165		X	
15		X		65			X	115		X		166			X
16		X		66			X	116			X	167		X	
17			X	67		X		117			X	168			X
18		X		68		X		118		X		169		X	
19			X	69		X		119		X		170			X
20		X		70		X		120		X		171		X	
21		X		71			X	121			X	172		X	
22		X		72		X		122		X		173	X		
23		X		73		X		123		X		174	X		
24		X		74		X		124		X		175	X		
25		X		75		X		125		X		176	X		
26		X		76		X		126			X	177	X		
27		X		77			X	127		X		178	X		
28			X	78			X	128		X		179	X		
29		X		79		X		129			X	180	X		
30		X		80			X	130		X		181	X		
31		X		81		X		132		X		182	X		
32		X		82			X	133			X	183	X		
33		X		83		X		134			X	185	X		
34			X	84		X		135		X		186	X		
35			X	85		X		136		X		187	X		
36		X		86		X		137		X		188	X		
37		X		87			X	138			X	189	X		
38		X		88			X	139		X		190	X		
39		X		89		X		140			X	191	X		
40		X		90		X		141		X		192	X		
41		X		91		X		142			X	193	X		
42		X		92		X		143		X		194	X		
43		X		93		X		144		X		195	X		
44			X	94			X	145		X		196	X		
45		X		95		X		146		X		197		X	
46		X		96		X		147		X		198		X	
47			X	97		X		148		X		199		X	
48			X	98		X		149		X		200		X	
49		X		99			X	150		X					
50			X	100			X	151			X				

B Written Instructions to the Assessors

Today's search systems take a question and return a list of documents likely to contain an answer to the question. The user of the system must then read the documents to find the desired answer within them, if it's there. This can be a very tedious, time-consuming process, and frustrating process.

It would be better if the system returned smaller pieces of text—a few words or a sentence or at most a paragraph believed to contain the answer. Then the user would have less reading to do in order to see if any of the pieces contained an answer.

Such improved systems exist today in experimental versions. In order to know how good a job such improved search systems are doing we need to judge whether, given a question, the systems return pieces of text that are responsive (i.e., you can recognize the answer in the piece) to the question. Your task will be to make these judgments.

For each question you will be given several pieces of text varying in size from a few words to a paragraph. The experimental search systems returned these pieces of text in response to this question. Your job is to decide whether each piece of text really contains some words that answer the question. Here is how you should proceed.

For each question:

1. Read the question carefully, then
2. Find the answer by skimming through the document and answer strings. It may be the case that no one retrieved the answer. If this happens, see Dawn.
3. For each answer string, read the piece of text and judge whether it contains a valid answer to the question.
 - If the answer string is the answer, judge it correct (yes).
 - If the answer string contains the answer plus supporting text, judge it correct (yes).
 - If the answer string contains the answer plus miscellaneous other stuff, judge it correct (yes).
 - If the answer string contains the answer plus other text that interferes with recognizing the answer, then you should decide how much interference there is, but it is probably incorrect.
 - If the answer string does not contain the answer, then judge it incorrect (no).

Notes:

- It's possible that there may be more than one answer. Check the document to see whether an answer string contains a valid answer.
- Judge the answer in the context of the document – even if the document gives an answer that you believe is wrong, judge the answer string on the basis of what the document says anyway.
- It is up to the assessor to decide if “partial answers” are responsive. Example: Last name only may be acceptable for “who” questions.

C Assessor Agreement per Question

Table 4: Number of answer strings judged (J); number of answers that were judged differently by different assessors (D); number of majority opinions overruled in adjudication (O); and overlap of the correct sets across assessors (OL). The overlap is undefined if no answers are judged as correct.

Qid	J	D	O	OL	Qid	J	D	O	OL	Qid	J	D	O	OL	Qid	J	D	O	OL
1	189	17	3	.66	51	200	12	1	.83	101	199	1	0	.94	152	200	0	0	1.0
2	175	0	0	1.0	52	197	7	0	.78	102	202	31	1	0.0	153	200	7	3	.22
3	195	47	2	.49	53	202	3	1	.88	103	197	22	1	.49	154	196	15	1	.53
4	189	1	1	.89	54	189	28	1	0.0	104	184	1	1	.86	155	193	42	2	.43
5	202	1	0	.97	55	185	23	3	.28	105	202	11	0	.27	156	198	16	0	0.0
6	185	9	0	.64	56	192	1	0	.94	106	191	14	2	.62	157	197	17	4	.77
7	191	6	1	.70	57	196	20	2	.50	107	188	3	0	.77	158	194	3	0	.87
8	191	19	0	.10	58	183	14	8	.85	108	203	27	10	0.0	159	176	2	0	.90
9	185	2	0	.80	59	202	19	2	.60	109	198	12	3	.43	160	187	26	2	.62
10	180	9	0	.70	60	194	12	4	.64	110	195	15	4	.75	161	175	17	11	.48
11	200	0	0	1.0	61	183	4	2	.73	111	188	5	0	.62	162	202	11	5	.82
12	197	0	0	1.0	62	186	8	0	.73	112	205	9	0	.90	163	199	31	12	.61
13	198	4	1	.85	63	195	3	0	.97	113	199	5	0	.77	164	195	30	14	.29
14	191	37	1	.31	64	194	1	0	.95	114	184	1	0	.92	165	182	1	0	.89
15	187	4	0	.73	65	187	30	15	.35	115	179	18	2	.10	166	193	7	0	.30
16	189	1	0	.95	66	178	1	0	.96	116	199	12	0	0.0	167	195	8	2	0.0
17	197	2	1	.87	67	192	6	1	.40	117	193	32	4	.53	168	202	4	0	.87
18	190	64	0	.36	68	174	3	0	.90	118	177	13	1	.61	169	198	0	0	—
19	192	4	0	.92	69	191	4	1	.60	119	187	4	2	.64	170	197	15	1	.83
20	172	10	0	.44	70	199	11	0	.83	120	194	13	1	.64	171	195	2	2	.94
21	196	1	0	.90	71	197	7	0	.78	121	202	11	1	.74	172	198	18	0	.76
22	184	1	0	.89	72	185	0	0	1.0	122	191	1	0	.97	173	199	26	2	0.0
23	186	0	0	1.0	73	207	69	1	.10	123	197	1	0	.94	174	175	12	0	.50
24	194	37	2	.54	74	198	20	0	.50	124	192	30	3	.47	175	176	4	0	.56
25	190	4	1	.75	75	194	9	1	.47	125	189	0	0	—	176	195	0	0	1.0
26	195	1	1	.96	76	196	26	0	.50	126	196	20	1	0.0	177	196	4	0	.87
27	183	11	0	.45	77	196	0	0	—	127	186	10	2	.72	178	204	20	5	.76
28	190	10	0	.84	78	190	24	14	.43	128	201	15	0	.73	179	181	12	7	.66
29	201	12	2	.64	79	189	17	0	.39	129	197	1	0	.96	180	199	13	1	.85
30	182	4	1	.88	80	186	1	0	.99	130	192	3	0	.89	181	195	0	0	—
31	198	6	0	.79	81	194	1	0	.95	132	197	22	1	.64	182	186	4	1	.20
32	191	3	3	.67	82	195	7	1	.93	133	173	0	0	1.0	183	195	5	0	.86
33	198	10	3	.70	83	195	13	0	.68	134	185	34	0	.52	185	184	8	0	.81
34	176	5	0	.75	84	187	0	0	1.0	135	198	6	0	.50	186	175	14	3	.46
35	190	5	0	.90	85	187	34	1	.65	136	193	0	0	—	187	178	8	2	.62
36	193	2	0	.83	86	193	10	2	.57	137	184	6	3	.85	188	193	18	0	.36
37	192	6	2	.86	87	191	2	2	.60	138	202	3	0	.90	189	200	10	5	.17
38	193	1	1	.96	88	195	6	0	.82	139	189	24	0	0.0	190	193	16	5	.70
39	193	26	7	.73	89	192	5	0	.29	140	183	4	0	.93	191	194	10	4	.60
40	197	8	0	.85	90	182	7	1	.90	141	184	5	0	.44	192	190	30	4	.29
41	196	19	1	.59	91	205	11	6	.45	142	188	0	0	1.0	193	185	20	14	.09
42	189	12	1	.78	92	187	7	0	.67	143	198	18	18	.17	194	190	7	5	.63
43	198	7	0	.46	93	196	4	0	.71	144	192	32	1	.41	195	196	18	3	.57
44	199	0	0	1.0	94	197	0	0	1.0	145	177	16	3	.62	196	200	9	0	.40
45	185	42	0	.09	95	195	43	20	.16	146	197	6	1	0.0	197	191	5	0	.55
46	197	6	0	.86	96	184	2	0	.33	147	188	18	1	.84	198	169	14	0	.12
47	191	7	1	.61	97	192	12	3	0.0	148	196	3	0	.62	199	192	11	2	.76
48	185	6	2	.92	98	182	4	1	.64	149	203	10	4	.80	200	188	13	3	.50
49	193	23	8	.18	99	177	3	1	.93	150	191	2	1	.78					
50	197	4	2	.83	100	191	26	1	.65	151	190	1	0	.96					

The TREC Spoken Document Retrieval Track: A Success Story

John S. Garofolo, Cedric G. P. Auzanne, Ellen M. Voorhees

National Institute of Standards and Technology

100 Bureau Drive, Mail Stop 8940

Gaithersburg, MD 20899-8940

USA

{john.garofolo, cedric.auzanne, ellen.voorhees}@nist.gov

Abstract

This paper describes work within the NIST Text REtrieval Conference (TREC) over the last three years in designing and implementing evaluations of Spoken Document Retrieval (SDR) technology within a broadcast news domain. SDR involves the search and retrieval of excerpts from spoken audio recordings using a combination of automatic speech recognition and information retrieval technologies. The TREC SDR Track has provided an infrastructure for the development and evaluation of SDR technology and a common forum for the exchange of knowledge between the speech recognition and information retrieval research communities. The SDR Track can be declared a success in that it has provided objective, demonstrable proof that this technology can be successfully applied to realistic audio collections using a combination of existing technologies and that it can be objectively evaluated. The design and implementation of each of the SDR evaluations are presented and the results are summarized. Plans for the 2000 TREC SDR Track are presented and thoughts about how the track might evolve are discussed.

1.0 TREC

The National Institute of Standards and Technology sponsors an annual Text REtrieval Conference (TREC) that is designed to encourage research on text retrieval for realistic applications by providing large test collections, uniform scoring procedures, and a forum for organizations interested in comparing results (Voorhees, et al., 2000). The conference, however, is only the tip of the iceberg. TREC is primarily an evaluation-task-driven research program. Each TREC research task culminates in a common evaluation just prior to the conference. The results of the evaluations are published by NIST in the TREC workshop notebook and conference proceedings. The sites participating in the evaluations meet at TREC to discuss their approaches and evaluation results and plan for future TREC research tasks.

In recent years the conference has contained one main task and a set of additional tasks called tracks. The main task investigates the performance of systems that search a static set of documents using new questions. This task is similar to how a researcher might use a library---the collection is known but the questions likely to be asked are not known. The tracks focus research on problems related to the main task, such as retrieving documents written in a variety of languages using questions in a single language (cross-language retrieval), retrieving documents from very large (100GB) document collections, and retrieval performance with humans in the loop (interactive retrieval). Taken together, the tracks represent the majority of the research performed in the most recent TRECs, and they keep TREC a vibrant research program by encouraging research in new areas of information retrieval. The three most recent TRECs (TREC-6 – TREC-8) have also included a Spoken Document Retrieval (SDR) track.

2.0 Spoken Document Retrieval

The motivation for developing technology that can provide access to non-textual information is fairly obvious. Large multi-media collections are already being assembled. The explosive growth of the Internet has enabled access to a wealth of textual information. However, access to audio information, and specifically spoken audio archives is pitifully limited to audio which has been manually indexed or transcribed. It is true that commercial human-generated transcripts are now available for many radio and

television broadcasts, but a much greater body of spoken audio recordings (untranscribed legacy radio and television broadcasts, recordings of meetings and conferences, classes and seminars, etc.) remains virtually inaccessible. The TREC Spoken Document Retrieval (SDR) track has been created to begin to address these problems.

SDR provides content-based retrieval of excerpts from archives of recordings of speech. It was chosen as an area of interest for TREC because of its potential use in navigating large multi-media collections of the near future and because it was believed that the component speech recognition and information retrieval technologies would work well enough for usable SDR in some domains. SDR technology opens up the possibility of access to large stores of previously unsearchable audio archives and paves the way for the development of access technologies to multimedia collections containing audio, video, image, and other data formats. (Voorhees et. al., 1997a)

In practice, SDR is accomplished by using a combination of automatic speech recognition and information retrieval technologies. A speech recognizer is applied to an audio stream and generates a time-marked transcription of the speech. The transcription may be phone- or word-based in either a lattice (probability network), n-best list (multiple individual transcriptions), or more typically, a 1-best transcript (the most probable transcription as determined by the recognizer). The transcript is then indexed and searched by a retrieval system. The result returned for a query is a list of temporal pointers to the audio stream ordered by decreasing similarity between the content of the speech being pointed to and the query (Garofolo et al., 1997b). A typical SDR process is shown in Figure 1.

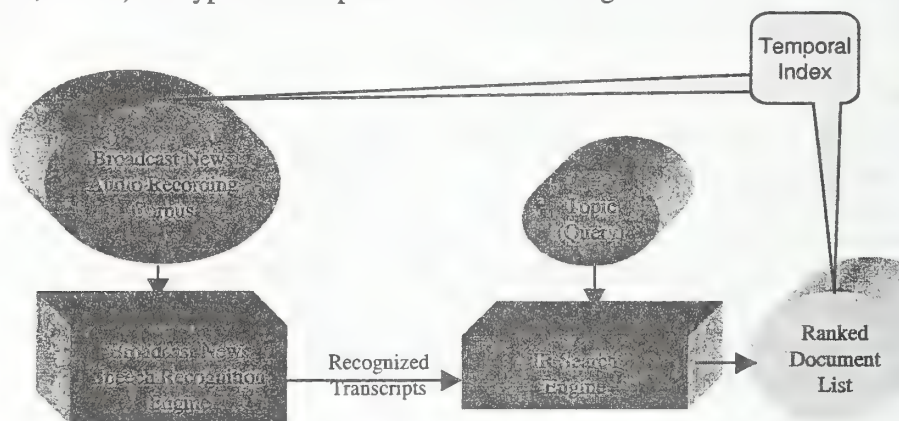


Figure 1: Typical SDR Process

3.0 TREC SDR Background

In 1996, an evaluation of retrieval using the output of an optical character recognizer (OCR) was run as a "confusion" track in TREC-5 to explore the effect of OCR errors on retrieval (Kantor, et al., 2000). This track showed that it was possible to implement and evaluate retrieval on "corrupted" text. After implementing this track, NIST and members of the TREC community thought it would be interesting to implement a similar experiment using automatic speech recognition (ASR).

During the 1996 TREC-5 workshop, researchers from NIST and the TREC community led by Karen Spärck Jones from the University of Cambridge met to discuss the possibility of applying information retrieval techniques to the output of speech recognizers. While the NIST Natural Language Processing and Information Retrieval Group had been supporting the evaluation of retrieval technologies under the auspices of TREC, the NIST Spoken Natural Language Processing Group had been working with the DARPA automatic speech recognition (ASR) community in evaluating speech recognition technology on

radio and television broadcast news. The broadcast news evaluation task had accelerated progress in the recognition of real data and it seemed that the technology was producing transcripts with reasonable enough accuracy for investigation of downstream application uses such as SDR. The DARPA ASR community also had access to a 100-hour corpus of broadcast news recordings collected by the Linguistic Data Consortium (LDC) for ASR training (Graff et al., 1996) that for the first time provided a data collection which might be sufficiently large for SDR.

The NIST Spoken Natural Language Processing Group and Natural Language Processing and Information Retrieval Group joined forces to develop a plan for the creation of a research track within TREC to investigate the new hybrid technology. The primary goal of the track would be to bring the speech and information retrieval communities together to promote the development of SDR technologies and to track progress in their development. The track would also foster research on the development of large-scale, near-real-time, continuous speech recognition technology as well as on retrieval technology that is robust in the face of input errors. More importantly, the track would provide a venue for investigating hybrid systems that may be more effective than simple stove-pipe combinations. Thus, the track would also encourage cooperation and synergy between groups with complementary speech recognition and information retrieval expertise.

4.0 TREC-6 SDR: Known Item Retrieval

4.1 Evaluation Design

The first year for the SDR Track was truly one of getting the speech and IR communities together and exploring the feasibility of implementing and evaluating SDR technology. Toward that end, the TREC-6 SDR evaluation was designed for easy entry and straight-forward implementation. Since it would be the first common evaluation of SDR technology, the evaluation itself was also considered to be experimental. While the main TREC task was focussing on ad-hoc retrieval of multiple relevant documents from single topics, we decided that the first SDR Track should employ a *known-item* retrieval task which simulates a user seeking a particular, half-remembered document in a collection. The goal in a known-item retrieval task is to generate a single correct document for each topic rather than a set of relevant topics as in an ad-hoc task. This approach simplified the topic selection process and eliminated the need for expensive relevance assessments. It was also thought at the time that an SDR ad-hoc retrieval task might produce results too poor to evaluate and would discourage participation (Voorhees, et al., 1997a).

Early on we decided that the evaluation should measure not only the end-to-end effectiveness of SDR systems, but the individual ASR and IR components as well. To that end, the evaluation included several complementary runs – all using the same set of topics, but with different sets of transcriptions of the broadcast news recordings in the test collection:

- Reference** retrieval using “perfect”¹ human-transcribed reference transcriptions

- Baseline** retrieval using “given” IBM ASR-generated transcriptions

- Speech** retrieval using the recordings themselves, requiring both ASR and IR components

The Reference run permitted the evaluation of the overall effectiveness of the retrieval algorithms on a spoken language collection while removing ASR as a factor. Likewise, the Baseline condition permitted the comparison of the effectiveness of retrieval algorithms on the same errorful ASR-produced transcripts. Finally, the Speech run permitted the evaluation of full end-to-end SDR performance.

The Reference transcripts which were contributed by the LDC were formatted in Hub-4-style UTF format files – one for each broadcast (Garofolo, et al., 1997a). The Baseline recognizer transcripts were contributed by IBM (Dharanipragada et al., 1998). The Baseline and shared recognized transcripts were

¹ Human transcripts are not actually perfect. Hub-4 training quality transcripts are generally believed to contain 3 – 4% WER.

stored in SGML-formatted files which included story boundaries and a record for each word including start and end times. The broadcast recordings were digitally sampled (16-bit samples, linear-PCM encoded, 16-KHz. sampling rate) using a single monophonic channel and stored in NIST SPHERE-formatted files.

This componentized approach served two purposes: First, it allowed different ASR and IR sites to join together to create pipelined systems in which the components could be mixed, matched, and separately evaluated. It also permitted retrieval sites without access to ASR systems to participate in a limited way by implementing only the Reference and Baseline retrieval tasks. The participation level for sites implementing both recognition and retrieval was deemed *Full SDR* and the participation level for sites implementing retrieval only was deemed *Quasi-SDR*. Although artificial, to simplify implementation and evaluation, sites would be given human-annotated story boundaries with story ID's for all test conditions. This permitted a simplified document-based approach to implementation and evaluation.

NIST developed 47 test topics – half designed by the NIST NLP Group to exercise classic IR challenges. The other half were designed by the SNLP Group to exercise challenges in the speech recognition part of the problem. Half of the “speech” topics were designed to target stories with “easy-to-recognize” speech (scripted speech recorded in studio conditions with native speakers and no noise or music in the background). The other half of the speech topics were designed to target stories with “difficult-to-recognize” speech (unscripted speech, speech over telephone channels, non-native speakers, and speech with noise or music in the background). The variety of topics would permit us to examine in more detail the effect of speech recognition accuracy on retrieval performance.

We found several important differences between broadcast news stories and document-based IR collections. First, the broadcast news stories were extremely short with regard to number of words. The TREC-6 SDR collection had an average number of 276 words per story with most stories containing 100 words or less. Full-text IR collections tend to have documents with many more words – usually an order of magnitude larger. Further about 1/3 of the stories in the SDR collection were annotated as “filler” -- non-topical transitional material. We filtered the collection to remove commercials, sports summaries, weather reports, and untranscribed stories. However, we decided to leave the filler segments in the test collection to keep it as large as possible. The final filtered broadcast news collection had only 1,451 stories. Although the collection represented a sizable corpus for speech recognition (previous test corpora were less than 3 hours), it was pitifully small for retrieval testing – at least 2 orders of magnitude smaller than current IR test collections.

The test specifications and documentation for the TREC-6 SDR track are archived at <http://www.nist.gov/speech/sdr97.txt>.

4.2 Test Results

The test participants were given 3 months to complete the evaluation. Thirteen sites or site combinations participated in the first SDR Track. Nine of these performed Full SDR: AT&T, Carnegie Mellon University, Claritech (with CMU ASR), ETH Zurich, Glasgow University (with Sheffield University ASR), IBM, Royal Melbourne Institute of Technology, Sheffield University, and University of Massachusetts (with Dragon Systems ASR). The remaining 4 sites performed Quasi SDR: City University of London, Dublin City University, National Security Agency, and University of Maryland. (See TREC-6 SDR participant papers)

Since the goal of the track was to evaluate retrieval performance, there was no formal evaluation of recognition performance. However, Full SDR sites were encouraged to submit their 1-best transcripts so that NIST could examine the relationship between recognition performance and retrieval accuracy. The

word error rate for the IBM Baseline recognizer was 50.0% (Dharanipragada et al., 1998). The mean story word error rate was a bit lower at 40%. The mean story word error rate for the other measured recognizers fell between 35% and 40%. These error rates were substantially higher than those obtained in the Hub-4 ASR tests. This difference was primarily due to three factors: The transcriptions used for scoring SDR ASR performance were created as ASR training material and had not been put through the rigorous verification that NIST employs for its Hub-4 evaluation test data. Likewise, a generic SCLITE orthographic mapping file was used. The orthographic mapping file maps alternate representations of certain words and contractions to a common format prior to scoring. A custom version of this file is created for each Hub-4 test set to minimize the number of alternative representation confusion errors. Finally, in order to process the 50-hour collection, several sites chose to use faster, less accurate recognizers than were used in the Hub-4 tests.

Initially, we believed that the retrieval results for the SDR Track would be quite poor. Therefore, we devised scoring metrics such as *Mean Rank When Found* and *Mean Reciprocal Rank* which gave systems partial credit for finding target stories at lower ranks (Voorhees, et al, 1997a). However, we were happily surprised to find that the systems performed quite well. So well, in fact, that we chose to use *Percent Retrieved at Rank 1* as our primary metric (Garofolo, et al, 1997b). Retrieval rates were very high for the Reference transcript condition and most sites showed only a small degradation for retrieval using their own recognizers. There was generally higher degradation in retrieval using the Baseline recognizer transcripts due to its high error rate and high number of *out-of-vocabulary* (OOV) words. The results of the evaluation for all three retrieval conditions are shown in Figure 2.

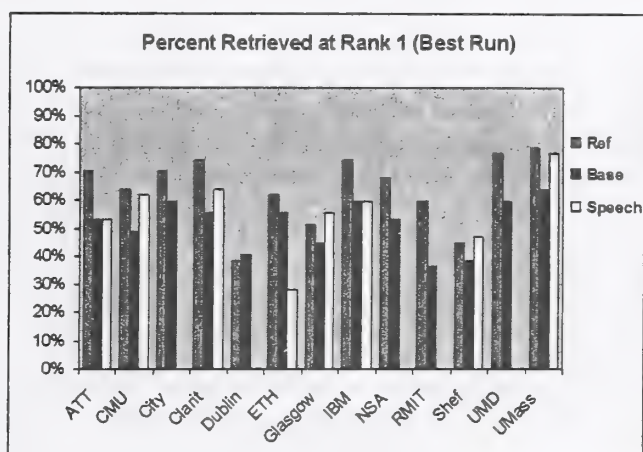


Figure 2: TREC-6 SDR Retrieval rate at rank 1 for all systems and modes (best run)

For Percent Retrieved at Rank 1, the best performance for all three test conditions was achieved by the University of Massachusetts System (with Dragon Systems recognition for Full SDR) which obtained a retrieval rate of 78.7% for the Reference condition, 63.8% for the Baseline recognizer condition, and 76.6% for the Speech condition (Allan et al, 1997). In fact, the UMass system missed only one more topic on the Speech condition than it did on the Reference condition.

An analysis of errors across systems for particular topics (Figure 3) showed that, in general, the "Easy to Recognize" topic set yielded the best performance for all 3 evaluation conditions while the "Difficult to Recognize" topic set yielded substantially degraded performance. However, the "Difficult Query" topic subset yielded even greater performance degradation. It is interesting to note that systems also had difficulty in retrieving stories for the "Difficult to Recognize" topic subset from the Reference transcriptions – an indication that factors in transcribed speech other than recognition errors might influence retrieval performance. However, there was far too much variance from the topic effect to make any sweeping conclusions.

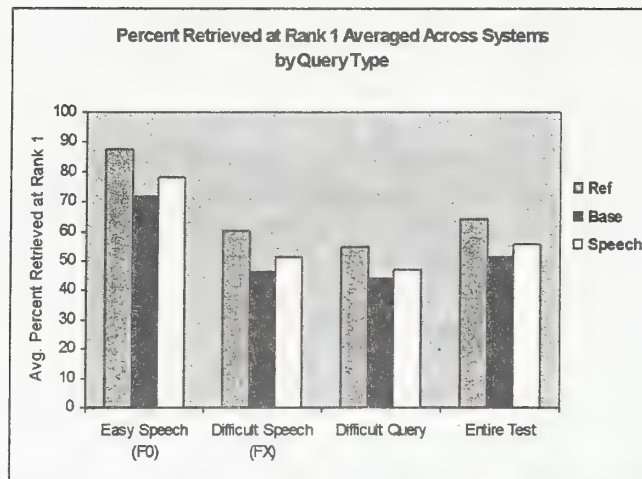


Figure 3 : TREC-6 SDR Percent Retrieval at Rank 1 averaged across systems by topic subset

To further examine the effect of recognition error rate on retrieval, we examined performance using the Baseline recognizer results. For each topic, we sorted the mean rank at which the retrieval systems found the target story against the word error rate for that story (Figure 4). The sorting appears to show an increasing trend toward poorer retrieval performance as recognition errors increase.

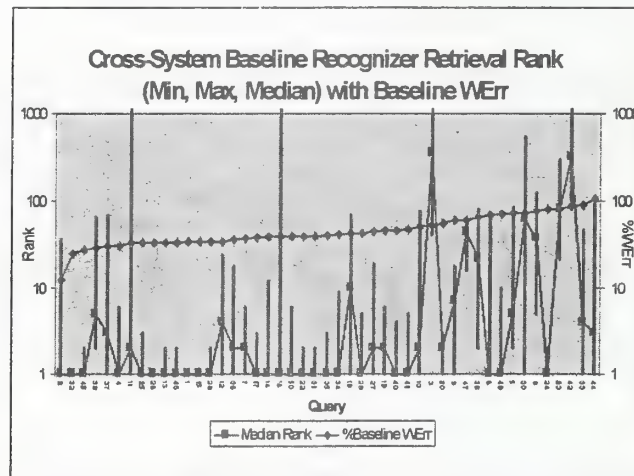


Figure 4 : TREC-6 Baseline condition mean retrieval rank sorted by Baseline Recognizer story word error rate

Interestingly, the same plot for retrieval for the Reference transcripts shows a similar trend (Figure 5) indicating that stories that are difficult to recognize may also be innately difficult to retrieve – even when recognized perfectly. One hypothesis is that the complexity of the language within the more difficult-to-recognize stories is greater than that of the more easy-to-recognize stories.

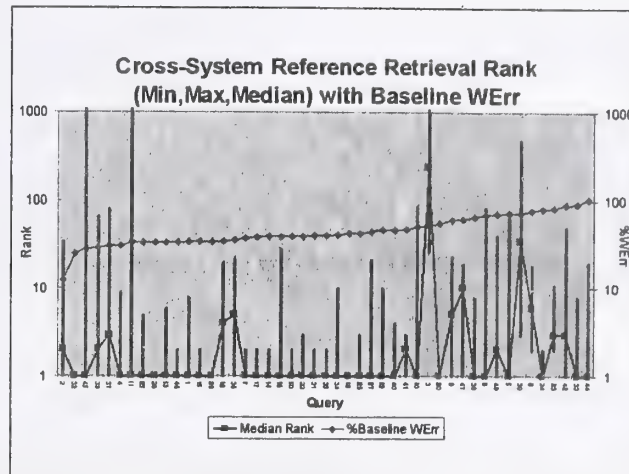


Figure 5 : TREC-6 Reference condition mean retrieval rank sorted by Baseline Recognizer story word error rate

A statistical analysis of variance showed that we had too little data to eliminate a large proportion of confounding unexplained factors (Garofolo, et al., 1997b). A future evaluation which would provide multiple recognizer transcript sets which all retrieval sites would run against would help to clarify the relationship between recognition and retrieval performance.

4.3 Conclusions

The first SDR evaluation showed us that we could successfully implement an evaluation of SDR technology and that existing component technologies worked well on a known-item task with a small audio collection. However, the test participants all agreed that the test collection would have to be enlarged by at least an order of magnitude before any “real” performance issues would surface. It was also agreed that the known-item task provided insufficient evaluation granularity. For this evaluation, it seemed that retrieval performance played a much more significant role in overall SDR performance than recognition performance. However, it was difficult to make any conclusions given the limited evaluation paradigm and collection.

5.0 TREC-7 SDR : Ad Hoc Retrieval

5.1 Evaluation Design

In 1998, for TREC-7, we set out to address some of the inadequacies in the TREC-6 SDR Track. We still did not have access to a large enough audio collection for true retrieval evaluation, but we were able to double the size of the SDR collection using an additional broadcast news corpus collected by the LDC for Hub-4 ASR training. More importantly, though, we decided to give up the known item retrieval paradigm and implement a classic TREC ad-hoc retrieval task.

In an ad hoc retrieval test, systems are posed with topics and attempt to return a list of documents ranked by decreasing similarity to the topic. The documents are then evaluated for relevance by a team of human assessors. In TREC, to keep the evaluation tractable, NIST pools the top N documents output by all of the evaluated systems and judges only those documents. Therefore, systems get evaluated over all documents, but only some documents are judged. Although not exhaustive, this approach assumes that with enough different systems, all of the relevant documents will be included in the pool. The traditional TREC ad-hoc track provided several forms of information for each topic: A title, a short query form -- usually a single sentence or phrase, and a descriptive narrative giving rules for judging relevance. Given

the limited size of the SDR collection, we decided to simplify the SDR topics to a single short form. We also required that all runs had to be fully automatic.

The TREC-7 SDR test collection contained 87 hours of audio with 2,866 usable stories after filtering and a similar mean and median story length as compared to the TREC-6 collection. As in TREC-6, participants were given human-annotated story boundaries and story IDs. This removed story-boundary detection from the technical challenge, but permitted NIST to use the standard TREC document-based TREC_EVAL scoring software to evaluate the results of the test. A team of 3 NIST TREC assessors created 23 test topics (averaging 14.7 words in length) for the collection. The following are two of the test topics they created:

Find reports of fatal air crashes. (Topic 62)

What economic developments have occurred in Hong Kong since its incorporation in the Chinese People's Republic? (Topic 63)

To more accurately examine the effect of recognition performance on retrieval, we decided to add a new optional evaluation condition, *Cross Recognizer Retrieval*, in which retrieval systems would run on other sites' recognized transcripts. This would permit us to more tightly control for the recognizer effect in our analyses as well as provide us with more information regarding the relationship between recognizer performance and retrieval performance. We therefore encouraged all sites running 1-best recognition to submit their recognizer transcripts to NIST for sharing with other participants. To permit sites to explore the effect of using different recognizers, we permitted each Full SDR site to run retrieval on both a primary (S1) and secondary (S2) recognizer.

For the Baseline recognizer, NIST created a local instantiation of the Carnegie Mellon University SPHINX-III recognizer. Since SPHINX-III ran in nearly 200 times real time on NIST's UNIX-based workstations, NIST realized that it would take nearly two years of computation to complete a single recognition pass over the 87-hour collection. NIST learned of inexpensive clusters of PC-LINUX-based systems being used by NASA in its BEOWULF project (BEOWULF, 1997) and set out to create a cluster-based recognition system. The final system incorporated a scheduling server and 40 computational nodes. Given the cluster's enormous computational power, to further enrich the spectrum of recognizers in the evaluation, NIST chose to create two Baseline recognizer transcript sets. One set (B1) was created using an "optimal" version of the SPHINX recognizer and benchmarked at 27.1% word error rate on the Hub-4 '97 test set (Pallett, et al., 1998) and at 33.8% on the SDR test collection. This enabled us to for the first time benchmark the difference in performance for the same recognizer running both Hub-4 and SDR ASR tests. A second set (B2) was created using lowered pruning thresholds and benchmarked at 46.6% word error rate for the SDR collection.

As in TREC-6, Full SDR sites were required to implement the Reference, Baseline, and Speech input retrieval conditions and the Quasi SDR sites were required to implement only the Reference and Baseline retrieval conditions.

The test specifications and documentation for the TREC-7 SDR track are archived at <http://www.nist.gov/speech/sdr98/sdr98.htm>.

5.2 Test Results

The TREC-7 SDR participants were given 4 months to implement the recognition portion of the task. They were then given one month to implement the required retrieval tasks and an additional month to

implement the optional Cross Recognizer retrieval task. The sites were not restricted in the hardware or number of processors they could apply in implementing the evaluation.

Eleven sites or site combinations participated in the second SDR Track. Eight of these performed Full SDR: AT&T [ATT], Carnegie Mellon University Group 1 [CMU1], University of Cambridge [CUHTK], DERA [DERA], Royal Melbourne Institute of Technology [MDS], Sheffield University [SHEF], The Netherlands Organization - TPD TU-Delft [TNO], and University of Massachusetts (with Dragon Systems ASR) [UMass]. The remaining 3 sites performed Quasi SDR : Carnegie Mellon University Group 2 [CMU2], National Security Agency [NSA], and the University of Maryland [UMD]. (See TREC-7 SDR participant papers)

In addition to the two NIST Baseline recognizers, 1-best transcripts for 6 additional recognizers were submitted to NIST for scoring and sharing in the Cross Recognizer retrieval condition. The recognizers covered a wide range of error rates and provided a spectrum of material for the Cross Recognizer retrieval condition. Figure 6 shows the word error rate and mean story word error rate for each of the submitted recognizer transcripts.

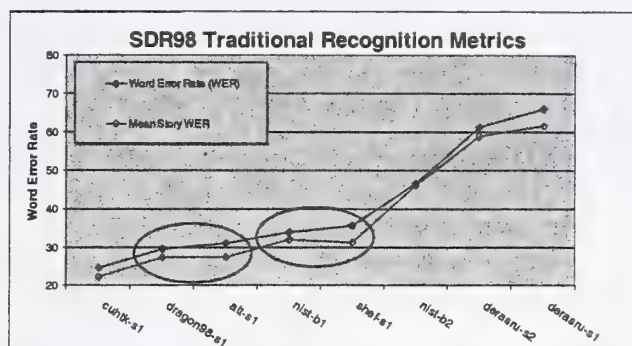


Figure 6: TREC-7 SDR Test set word error rate (WER) and mean story word error rate (SWER) for submitted recognized transcripts with cross-system significance at 95% for SWER

The best recognition results were obtained by the University of Cambridge HTK recognition system with a 24.6% test set word error rate and a 22.2% mean story word error rate (Johnson, et al., 1998). The circled mean story word error rate points were not considered to have statistically different performance. While the SDR ASR error rates were still significantly higher than Hub-4, in general, error rates were significantly improved from the previous year – even at the faster speeds required to recognize the larger test collection.

Each retrieval run was required to produce a rank-ordered list of the ID's for the top 1000 stories for each topic. The top 100 IDs from each of these lists were then merged to create the pools for human assessment. The 3 TREC assessors read the reference transcriptions for each of the topic pool stories to evaluate the stories for relevance. All of the retrieval runs were then scored using the standard TREC_EVAL text retrieval scoring software. As in other TREC ad hoc tasks, the primary retrieval metric for the SDR evaluation was mean average precision (MAP) which is the mean of the average precision scores for each of the topics in the run. The average precision is equivalent to the area underneath the uninterpolated recall-precision graph (Voorhees, et al., 1998).

In all, the TREC-7 SDR Track contained 6 retrieval conditions :

- Reference (R1): retrieval using Human (closed-caption-quality) reference transcripts
- Baseline-1 (B1): retrieval using NIST (CMU SPHINX) ASR transcripts
- Baseline-2 (B2): retrieval using NIST (CMU SPHINX) "sub-optimal" ASR transcripts

Speech-1 (S1): retrieval using participant's own recognizer

Speech-2 (S2): retrieval using participant's own secondary recognizer

Cross Recognizer (CR) : retrieval using other participants' recognizer transcripts

The results for each of the required test conditions: Reference (R1), Baseline-1 (B1), Baseline-2 (B2), Speech-1 (S1) and Speech-2 (S2) are shown in Figure 7. Full SDR participants were required to implement the R1, B1, B2, and S1 retrieval conditions. Quasi SDR participants were required to implement the R1, B1, and B2 retrieval conditions.

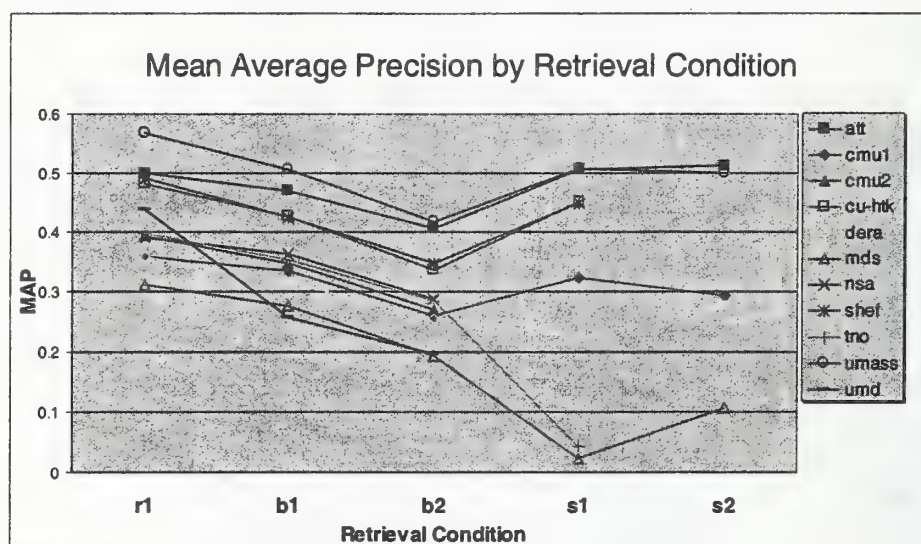


Figure-7: TREC-7 SDR Mean Average Precision (MAP) for required retrieval conditions

For all retrieval conditions except S2, the University of Massachusetts system (Allan, et al., 1998) achieved the best mean average precision. Most systems performed surprisingly well for the recognizer-based conditions. Even more surprising, AT&T's S2 run (the best recognizer-based run in the evaluation) outperformed its R1 run. AT&T attributed this excellent performance to a new approach they implemented for document expansion using contemporaneous newswire texts which they employed for their S1/S2 runs but not for their R1 run (Singhal, et al., 1998).

The most interesting condition for TREC-7 SDR was the cross recognizer retrieval (CR) condition in which participating systems ran retrieval on the 6 submitted recognizer-produced transcript sets in addition to the human Reference and B1/B2 recognizer transcript sets. This experiment gave us 9 recognition/retrieval data points to examine the effect of recognition performance on retrieval performance. Four sites (University of Cambridge, DERA, Royal Melbourne Institute of Technology [MDS], and Sheffield University) participated in the CR experiment. Using the mean story word error rate (SWER) ASR metric and the mean average precision (MAP) retrieval metric, we plotted the recognition/retrieval performance curve for each of the four systems (Figure 8).

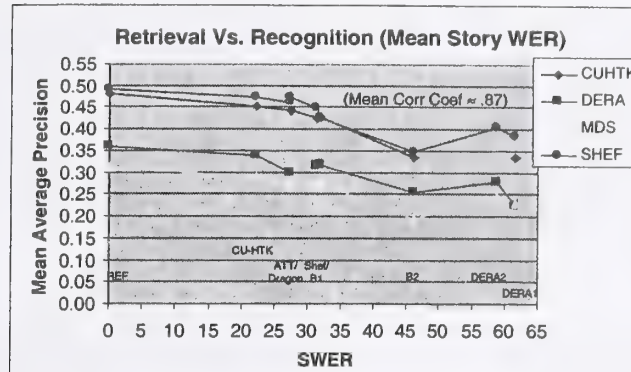


Figure 8: TREC-7 SDR Cross Recognizer results: mean average precision vs. mean story word error rate

The figure shows a gentle, but fairly linear drop-off in MAP for recognition transcripts with increasing SWER. We calculated the correlation coefficient for the metrics to determine how well SWER correlated with retrieval performance. The average correlation coefficient for the 4 systems was .87 – a significant correlation.

We explored several other word-error-rate-based metrics to see if we could find an even better predictor for retrieval performance. Our hypothesis was that such a metric would be useful in developing ASR systems for retrieval purposes. We explored metrics which used IR methods to filter out unimportant words for retrieval: *stop-word-filtered word error rate* and *stemmed stop-word-filtered word error rate* (Garofolo, et al., 1998). Surprisingly, however, these metrics turned out to be only slightly more correlated with mean average precision than word error rate. Other effective approaches to IR-customized ASR scoring using the TREC SDR data have been explored and reported by Johnson (1999) and Singhal (1999).

While we were implementing the TREC-7 SDR track, we were also administering a first evaluation in Named Entity (NE) tagging using broadcast news. The NE evaluation involved identification of people, locations, and organizations in broadcast news ASR transcripts (Przybocki, et al., 1999). To our fortune, GTE/BBN had hand-annotated the same data we used in the SDR evaluation with Named Entity tags (Miller, et al., 1999). Our hypothesis was that these named entities would identify most of the key content-carrying words in our spoken documents and that if we focussed our ASR metric on these words, we would obtain a better predictor of retrieval performance than by measuring the error rate of all words. We re-scored the ASR systems using the named entity word error rate and plotted the ASR metric against the mean average precision as we had done with mean story word error rate (Figure 9).

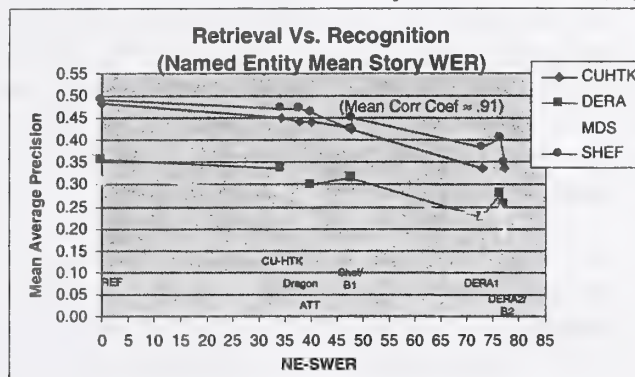


Figure 9: TREC-7 SDR Cross Recognizer results: mean average precision vs. named entity mean story word error rate

The plot showed a nearly linear relationship between named entity ASR performance and retrieval performance with a mean correlation coefficient of .91 across the systems. Most significantly, the plot more accurately positioned the problematic NIST B2 recognizer which had systematically-increased errors in longer (probably more-content-carrying) words. For all the systems, the named-entity-based metric showed a higher correlation with mean average precision than word error rate alone (Garofolo, et al., 1998). Other things being equal, this finding tells us that an ASR system which recognizes named entities most accurately will provide the best input for retrieval.

5.3 Conclusions

For TREC-7, we learned that we could successfully implement and evaluate an ad hoc SDR task. With the new Cross Recognizer condition, we were able to begin to investigate the relationship between recognition performance and retrieval performance. We found a near-linear relationship between word error rate and mean average precision and we found that recognition content-word-based word error metrics such as named entity word error rate provided even better predictors of retrieval performance than word error rate alone. Although twice the size of it's predecessor in number of stories, our 87-hour collection was still too far too small to make conclusions about the usefulness of the technology. Further, we were still evaluating systems using artificial human-annotated story boundaries.

6.0 TREC-8 SDR : Large Audio Collection

6.1 Evaluation Design

In 1998, the Linguistic Data Consortium began collecting a large radio and television corpus for the DARPA Topic Detection and Tracking (TDT) program. In contrast to most TREC tracks², the TDT program, is concerned with detecting and processing information from a continuous stream as it occurs in an *online* manner (Fiscus, et al., 1999). The TDT-2 corpus, collected to support the TDT program in 1998-99, contains news recordings from ABC, CNN, Public Radio International, and the Voice of America. With the exception of the VOA broadcasts, which began in early March, these sources were sampled evenly over a 6-month period between January and June 1998. The corpus also contains a contemporaneous newswire corpus containing articles from the New York Times and Associated Press (Cieri, et al., 1999).

With it's time-sampled broadcast news sources and parallel text corpus, the 600-hour TDT-2 corpus was also almost perfectly suited for use in the SDR Track. Unfortunately, it had no high-quality human reference transcriptions – only “closed-caption” quality transcriptions. Since the transcription quality prevented us from reasonably evaluating recognition performance over the entire collection, we selected a 10-hour randomly-selected story subset of the collection for detailed transcription by the LDC. These high-quality transcripts would permit us to perform a sampled evaluation of the ASR performance. They also permitted us to evaluate the error rate in the closed-caption-quality transcriptions themselves which we found to have roughly 14.5% WER for television closed-caption sources and 7.5% WER for radio sources which had been quickly transcribed by commercial transcription services (Fisher, 1999). These error rates are significant and the television closed caption error rates approach the error rates for state-of-the-art broadcast news recognizers.

Several SDR participants were also Hub-4 participants and intended to use their Hub-4 ASR systems which contained training data from January 1998 (which overlapped with the first month of the TDT-2 corpus.) To eliminate the possibility of training/test cross-contamination, we eliminated the January data from the SDR collection. The final collection contained 557 hours of audio collected between February

² The TREC Filtering track works on an online retrieval task similar to TDT.

1, 1998 and June 30, 1998. The collection contained 21,754 stories – an order of magnitude larger than the 87-hour TREC-7 SDR collection.³

We believe that deployed SDR systems will operate in an archive search modality. The most efficient means to implement such a system is to employ *online recognition* (in which recognition is performed on a continuous basis as audio is recorded) and *retrospective retrieval* in which the entire collection is queried after it is formed. This is in contrast to a TDT-type system which performs *online retrieval* as the audio is recognized. In both modalities, recognition should use adaptation techniques to adjust to changes in the collection language over time. Traditional Hub-4-style broadcast news recognizers employed only static pre-trained language models. If such a recognizer was used in a real time-longitudinal application, the language in the news and the fixed language model used in the recognizer would diverge, resulting in increasing error rates over time. Such recognizers are incapable of recognizing new words – words likely to be important for retrieval. Conversely, given the computational expense of performing recognition, *retrospective recognition* at the time of retrieval is impossible for realistically large collections. So, in a real SDR application where audio would be recorded over many months or years, the recognizer would have to be re-trained periodically to accommodate changes in the language and new words. To support this modality, we defined an online recognition mode which supported the use of evolving “rolling” language models in which the recognition systems could be periodically retrained over the test epoch. Full SDR sites were permitted to use either a traditional pre-trained recognition system or a continuously adaptive recognition system which used the contemporaneous newswire text from days prior to the day being recognized for adaptation. Sites were free to choose whatever retraining period or strategy they liked as long as they didn’t “look ahead” in time as they performed recognition (Garofolo, et al., 1999).

Realizing that the CMU SPHINX recognizer was far too slow to recognize the TREC-8 collection, NIST set out to find a faster baseline recognizer. During 1998, NIST added a spoke to its Hub-4 broadcast news ASR evaluation in which systems had to run in 10 times real time or fast on a single processor. This spoke, dubbed *10Xrt*, encouraged the development of fast broadcast news recognizers which suffered little degradation in recognition accuracy over their 150Xrt+ cousins (Pallett, et al., 1999). GTE/BBN offered NIST a LINUX instantiation of their fast BYBLOS Rough ‘N Ready recognizer (which now operated at 4Xrt) to use as a baseline in the SDR and TDT tests (Kubala, et al., 2000). BBN also gave NIST a basic language modeling toolkit to work with. Given the computational power of NIST’s recognition cluster and the speed of the BBN recognizer, NIST set out to create 2 complementary baseline recognizer transcript sets. The first set (B1) used a traditional Hub-4 fixed language model. The B1 recognizer benchmarked at 24.7% WER on the Hub-4 ‘97 test set, 23.4% WER on the Hub-4 ‘98 test set, and 27.5% WER on the SDR-99 10-hour subset. NIST then created an adaptive “rolling” language model version (B2) that used the SDR contemporaneous newswire texts for periodic look-back language model training. Details regarding the B2 recognizer are provided in Auzanne, et al. (2000). The B2 system benchmarked at 26.7% WER on 10-hour SDR-99 subset. This difference in performance might seem insignificant. However, NIST statistical tests showed that it is significantly different than the B1 recognizer. Further, the small decrease in word error belies a more significant decrease in the out-of-vocabulary (OOV) rate of the recognizer. The OOV rate is the percentage of test set words which are not included in the recognizer’s vocabulary and which, therefore, can never be correctly recognized. The OOV rate for the fixed B1 recognizer was 2.54%. The OOV rate for the adaptive B2 recognizer was 1.97% -- a 22.4% relative improvement.

In addition to the Reference, Baseline, Speech, and Cross Recognizer retrieval conditions used in TREC-7, an optional story boundaries unknown (SU) condition was added for TREC-8. This condition

³ The difference in story density is explained by the large proportion of short CNN stories in the TREC-8 collection. The average story length in the TREC-8 collection is only 169 words.

permitted sites to explore SDR where they had to operate on whole broadcasts with no knowledge of human-annotated topical boundaries. This condition more accurately represented the real SDR application challenge. A new ad-hoc paradigm had to be created to support the SU condition since it was not document based as in previous evaluations. The natural unit for audio recordings is time rather than documents or words. Therefore, it was decided that SU systems would output a ranked list of time pointers. Given that the TDT program was already investigating technology for story segmentation, we did not want to require SDR systems to find the topical boundaries in the audio recordings. Rather, we decided to require them to emit only a single time pointing to a "hot spot" or mid-point of a topical section. This approach allowed us to map the emitted times to known stories and make use of our traditional document retrieval evaluation software. Thus, this approach focussed on a new and interesting problem while making use of the existing evaluation infrastructure and permitting some comparison between runs where story boundaries were known and runs where they weren't known. To keep the task clean, we required that Full SDR sites implementing the SU option would also be required to run their recognizers without knowledge of story boundaries. However, to make maximal use of the recognizers for the CR task, NIST devised a script to backfill the story boundaries into the SU ASR transcripts.

The new SU condition did pose some challenges for scoring. The biggest issue was how time pointers which mapped to the commercials, fillers, or the same stories should be treated. NIST decided to implement a mapping algorithm that would severely penalize the over-generation of time pointers. The pointers were first mapped to known story ID's. Duplicate story ID's, commercials, and fillers were then mapped to "dummy" ID's which would be automatically scored as non-relevant. The results were then scored as usual with TREC_EVAL. Since the story boundary known (SK) collection excluded commercials and other untranscribed segments that were included in the SU collection, direct comparisons between the two conditions would not be possible. However, this first SU evaluation would give us an idea of how difficult a technical challenge the SU condition would pose.

A team of 6 NIST assessors created the ad hoc topics for the evaluation. The goal in creating TREC topics is to devise topics with a few (but not too many) relevant documents in the collection to appropriately challenge retrieval systems. Prior to coming together at NIST, the assessors were told to review the news for the first half of 1998 and to come up with 10 possible topics each. The assessors then tested their putative topics against the Reference transcripts in the TREC-8 SDR collection using the NIST PRISE search engine. If a topic was found to retrieve 1 to 20 documents in the top 25, it was considered for inclusion in the test. Otherwise, the assessors were required to refine (broaden or narrow) or replace the topic to retrieve an appropriate number of relevant documents using PRISE. The assessors created approximately 60 topics. Topics with similar subjects or which were considered malformed were then excluded to yield the final test set containing 49 topics.

The test specifications and documentation for the TREC-8 SDR track are archived at <http://www.nist.gov/speech/sdr99/sdr99.htm>.

6.2 Test Results

The TREC-8 SDR participants were given approximately three and a half months to implement the recognition portion of the task and a month and a half to implement the required retrieval tasks. In order to give the participants the maximum possible amount of time to run recognition, the retrieval period overlapped the recognition period by one month. After the site's recognized transcripts were submitted to NIST, they were checked, filtered, formatted and distributed for the Cross Recognizer retrieval condition. The retrieval sites were then given 3 weeks to perform the CR task. Since NIST had limited time for assessment, only the pre-CR retrieval results were used to construct the pools for assessment, which took place in parallel with the CR test. As in TREC-7, the sites were not restricted in the hardware or number of processors they could apply in implementing the evaluation.

Ten sites or site combinations participated in the third SDR Track. Six of these performed Full SDR: AT&T [ATT], Carnegie Mellon University [CMU], University of Cambridge [CU-HTK], LIMSI [LIMSI], Sheffield University [SHEFFIELD], and Twenty One Consortium [TNO]. The remaining 4 sites performed Quasi SDR: The State University of NY at Buffalo [CEDAR], IBM [IBM], The Royal Melbourne Institute of Technology [MDS], and the University of Massachusetts [UMASS]. (See the TREC-8 participant publications)

In all, the TREC-8 SDR Track contained 11 retrieval conditions:

- Reference (R1): retrieval using Human (closed-caption-quality) reference transcripts
- Baseline-1 (B1): retrieval using NIST (BBN Byblos) fixed language model ASR transcripts
- Baseline-2 (B2): retrieval using NIST (BBN Byblos) adaptive language model ASR transcripts
- Speech-1 (S1): retrieval using site's own recognizer
- Speech-2 (S2): retrieval using site's own secondary recognizer
- Cross Recognizer (CR): retrieval using other site's recognizer transcripts
- Baseline-1 boundaries unknown (B1U)
- Baseline-2 boundaries unknown (B2U)
- Speech-1 boundaries unknown (S1U)
- Speech-2 boundaries unknown (S2U)
- Cross Recognizer boundaries unknown (CRU)

Full SDR sites were required to run the R1, B1, and S1 retrieval conditions. Quasi-SDR sites were required to run only the R1 and B1 retrieval conditions. The B2, CR and all story boundaries unknown conditions (*U) were optional.

We benchmarked the performance of the speech recognizer transcripts contributed by Full SDR sites for sharing in the Cross Recognizer condition using the 10-hour Hub-4-style transcribed subset of the SDR collection. The summary results are shown in Figure 10.

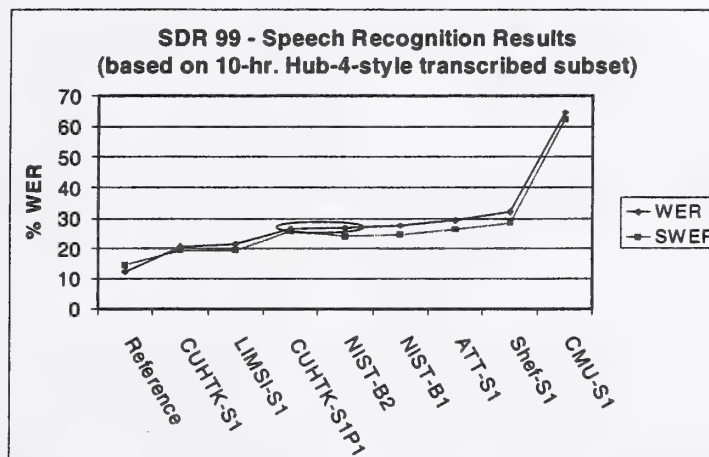


Figure 10 : TREC-8 SDR Speech Recognition Performance Results (Test Set Word Error Rate and Mean Story Word Error Rate) with cross-system significance for Word Error Rate

The word error rates were surprisingly low considering the enormous size of the test collection which was over 2 orders of magnitude larger than test sets used in Hub-4 ASR tests. The graph shows the results for both test-set word error rate and mean story word error rate. Most of the systems produced transcripts with word error rates of less than 30%. This is fairly impressive considering the speed at which the systems had to be run to process the large collection. It is also interesting to note that these scores are generally lower than the comparable scores from TREC-7 in which ASR systems were not run at such

fast speeds. The best ASR results were obtained by the University of Cambridge HTK recognizer with a 20.5% WER (Johnson, et al., TREC-8 1999). With the exception of the alternative first-pass-only Cambridge System and the NIST B2 system, none of the recognizer transcripts were found to be significantly similar in performance with respect to WER by the NIST statistical significance software. The figure also shows the results of scoring the original closed-caption-style Reference transcripts against the more scrupulously transcribed Hub-4-style transcripts.

As with the speech recognition performance, overall retrieval performance was quite good. As with all TREC ad hoc tests, there was quite a bit of variation in performance for particular topics. The following sample TREC-8 SDR test topics illustrate the variation:

Topic 105: *How and where is nuclear waste stored in New Mexico?*
(.85 average MAP across all systems/runs, 7 relevant stories).

Topic 117: *If we get more income, will we save more or spend more?*
(.34 average MAP across all systems/runs, 28 relevant stories)

Topic 94: *What percentage of the population is in prison in the U. S. A. and in the E. C. countries?*
(.01 average MAP across all systems/runs, 7 relevant stories)

Figure 11 shows the results for each of the non-Cross-Recognizer retrieval conditions. The best results for the Reference and Baseline-1 recognizer retrieval conditions were obtained by the AT&T system, with a MAP of .5598 and .5539 respectively (Singhal, et al., TREC-8 1999). The best result for the Speech input retrieval condition was obtained by the University of Cambridge system with a MAP of .5529 (Johnson, et al., TREC-8 1999). Sheffield University achieved the best performance for the Baseline and Speech input story boundary unknown conditions with a MAP of .4301 and .4250 respectively (Abberley et al., 1999).

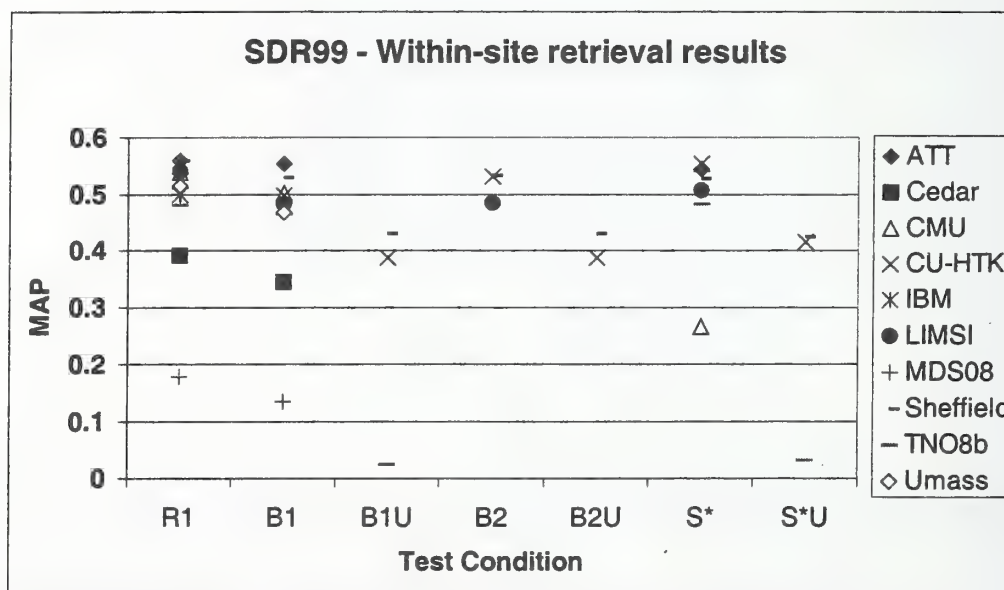


Figure 11: TREC-8 SDR Mean Average Precision (MAP) for required and non-cross-recognizer retrieval conditions

The individual test conditions were useful in contrasting the effect of binary variables such as human transcripts vs. ASR transcripts and story boundaries known vs. story boundaries unknown. However,

even more interesting results are found in the Cross-Recognizer retrieval conditions which contain multiple recognition performance/retrieval performance data points with which we can examine the effect of recognition performance on retrieval performance.

Four sites participated in the story boundaries known Cross-Recognizer (CR) retrieval condition: AT&T, University of Cambridge, LIMS, and Sheffield University. Each of these sites ran retrieval on the 8 sets of submitted recognizer transcripts. Adding the retrieval results for the closed-caption-quality Reference transcripts, this gives us 9 recognition/retrieval data points for each system. Figure 12 shows a graph of retrieval performance vs. recognition performance for the story boundaries known Cross-Recognizer retrieval condition. The CMU recognizer data point was removed since it was an extreme outlier. The graph shows that retrieval performance degrades very little for transcripts with increasing word error rates and that retrieval is fairly robust to recognition errors. Our hypothesis is that the redundancy of key words in the spoken documents permits the relevant documents to be retrieved – even when a substantial number of words are mis-recognized. For TREC-7, we assumed that this robustness was due to the small collection size and expected the recognition/retrieval performance drop-off to be much steeper for the larger TREC-8 collection. However, this does not appear to be the case. When we compare the average cross-system slope for the recognition/retrieval performance curve for TREC-7 and TREC-8, we find that they are almost identical (.0016 for TREC-8 vs. .0014 for TREC-7). Although the individual systems had different relative retrieval performance, all of the systems' slopes appears to be relatively flat. The AT&T system achieved the best CR performance and also had the most shallow recognition/retrieval performance slope (Singhal, et al., TREC-8 1999).

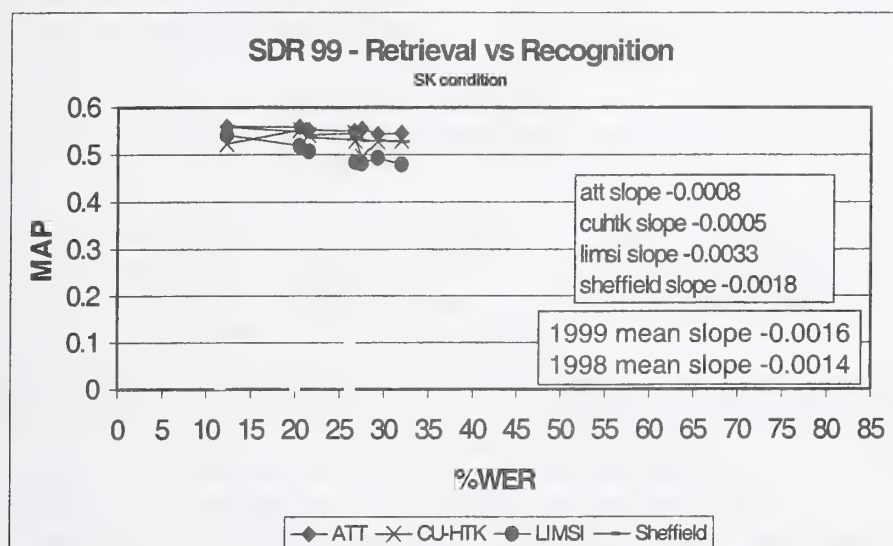
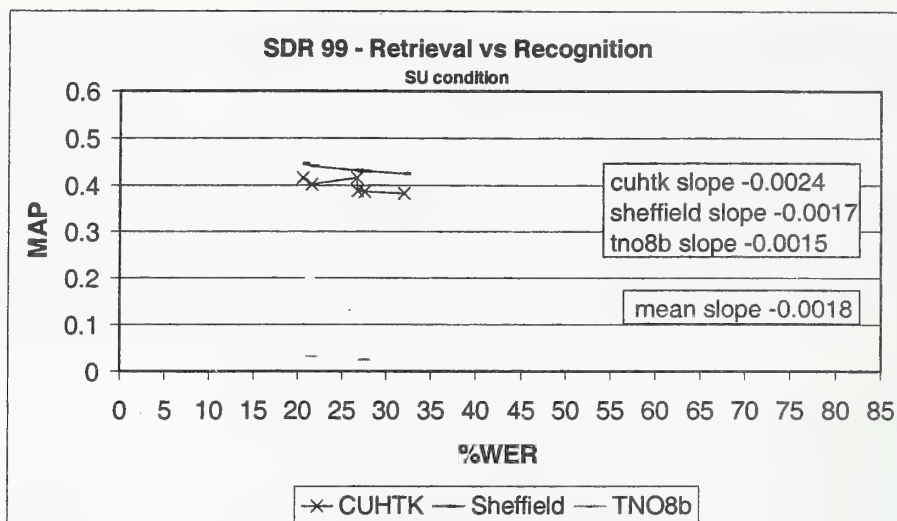


Figure 12 : TREC-8 SDR Story Boundaries Known Cross Recognizer Retrieval condition results showing Mean Average Precision vs. Word Error Rate

Three sites participated in the story boundaries unknown Cross Recognizer (CRU) retrieval condition: University of Cambridge, Sheffield University, and The Twenty One Consortium. The results of the CRU condition are shown in Figure 13.



The test specifications and documentation for the TREC-9 SDR track will be made available at <http://www.nist.gov/speech/sdr2000/sdr2000.htm>.

8.0 TREC SDR Track Conclusions and Future

The SDR Track has been an enormous success with regard to its primary goals of bringing the speech recognition and information retrieval research communities together to explore the feasibility of implementing and evaluating retrieval from spoken audio recordings. Certainly, we have shown that the technology can be implemented and evaluated for TREC known item and ad hoc tasks. We've also found that it can be implemented and evaluated for reasonably large audio collections and for conditions where story boundaries are unknown. In fact, progress has occurred so quickly, that one might conclude that SDR is a solved problem. However, there is still much useful non-lexical information to be harnessed from the audio signal. Further, while we have explored traditional text retrieval modalities using automatically transcribed speech, we haven't yet tackled such challenging problems as question answering or spoken queries in which the mis-recognition of a single word could cause catastrophic failure of the technology. In our traditional SDR task, the redundancy of words in the collection has protected us from truly facing these issues. Finally, there are still many more issues to explore and conquer with regard to the more general problem of multi-media information retrieval.

There has been much discussion regarding the future of the TREC SDR Track and several suggestions for future evaluations revolving around an audio-only domain have been circulated including passage retrieval, multi-lingual or cross-lingual SDR, SDR with question answering, interactive SDR, to name a few. However, most of these problems are already being tackled on a text-only basis within TREC and, with the possible exception of question answering, the additional information to be learned from them for audio collections might be somewhat limited. We now have a fairly good idea of the kinds of problems that ASR introduces for text retrieval and we can most likely model the behavior of other text retrieval domains using ASR without running full-blown evaluations.

It seems to us that the next challenge is, rather, a broadening to a true multi-media information retrieval (MMIR) domain which will require not only text retrieval and speech recognition, but video and still image processing as well.⁴ Further, these multi-media sources will come in many different forms which will need to be integrated and threaded. Such threading will no doubt require natural language processing and knowledge engineering. This is an enormous problem and will require collaboration among many different technology communities. For SDR, we brought together two research communities. MMIR will require the involvement of many more. Taken at once, this task seems virtually impossible. So, it will make sense to break it down into its constituent components or component combinations that can be incrementally integrated. Accordingly, we believe that several binary or ternary technology development and evaluation projects should be undertaken to explore the more tractable lower-level challenges before we undertake full MMIR. With this approach, core signal processing technologies such as speech recognition, speaker identification, face and object identification, scene tracking, etc. can be incrementally integrated with higher-level information processing technologies. Eventually, the capability to create robust multi-media information system technologies will emerge.

For next year, NIST is interested in creating a retrieval track that would begin to explore the information contained in the video signal. If a video corpus including audio is used, we can also begin to explore the integration of speech recognition and video processing into retrieval applications.

⁴ Actually, we've only scratched the surface of audio processing with speech recognition, since a great deal more information than words are encoded in the audio signal.

These new domains and integrated technologies will, of course, require the development of new evaluation methods, formats, and tools. This is perhaps one of the greatest challenges to overcome in developing a new technology research task. For each of the research tasks that NIST has created evaluation programs for, there has been significant and sometimes lengthy discussion and debate regarding the development of metrics and scoring protocols. Metrics which are taken for granted today, such as mean average precision and word error rate, were once hotbeds of discussion. Further, we will need to build not only component technology measures, but end-to-end system measures as multi-media systems technologies take shape. The possibilities are quite exciting, but there is much work to be done.

Acknowledgements

NIST work in the TREC SDR tracks was sponsored in part by the Defense Advanced Research Projects Agency (DARPA).

The authors would like to thank Karen Spärck Jones at the University of Cambridge for her guidance in the development of the SDR Track. We'd like to thank Donna Harman and David Pallett at NIST for their support for the SDR track and Vince Stanford at NIST for his help in implementing the baseline speech recognition systems. We'd like to thank Sue Johnson at the University of Cambridge for her help in refining the test specifications and evaluation protocols. We'd like to thank IBM for their contribution of the baseline speech recognizer transcripts for TREC-6 SDR, Carnegie Mellon University for their contribution of the SPHINX-III recognizer for TREC-7 SDR, and a special thanks to GTE/BBN for the contribution and support of their LINUX-based BYBLOS Rough 'N Ready fast recognizer for use in TREC-8 SDR. Finally, we'd like all the TREC SDR participants without whose participation this track would not have been such a success.

Disclaimer

Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by the National Institute of Standards and Technology nor does it imply that the products mentioned are necessarily the best available for the purpose.

Bibliographical References

- BEOWULF Project, NASA Center of Excellence in Space Data and Information Sciences, <http://cesdis.gsfc.nasa.gov/linux/beowulf/>, reviewed in 1997.
- Cieri, C., Graff, D., Liberman, M., Martey, N., Strassel, S, TDT-2 Text and Speech Corpus, Proc. 1999 DARPA Broadcast News Workshop, March 1999.
- Fiscus, J.G., Doddington, G., Garofolo, J.S., *NIST's 1998 Topic Detection and Tracking Evaluation*, Proc. 1999 DARPA Broadcast News Workshop, February 1999.
- Fisher, re: investigation of TDT-2 transcription error rates, personal conversation, 1999.
- Garofolo, J., Fiscus, J., and Fisher, W., *Design and preparation of the 1996 Hub-4 Broadcast News Benchmark Test Corpora*, Proc. DARPA Speech Recognition Workshop, February 1997.
- Garofolo, J., Voorhees, E., Stanford, V., and Spärck Jones, K., *TREC-6 1997 Spoken Document Retrieval Track Overview and Results*, Proc. TREC-6, 1997 and 1998 DARPA Speech Recognition Workshop, February 1998.
- Garofolo, J. S., Voorhees, E. M., Auzanne, C.G.P. , Stanford, V.M., Lund, B.A., *1998 TREC-7 Spoken Document Retrieval Task Overview and Results*, Proc. TREC-7, Nov. 1998.
- Garofolo, John S., Auzanne, Cedric G. P., Voorhees, Ellen M., *1999 Trec-8 Spoken Document Retrieval Track Overview and Results*, Proc. TREC-8, Nov. 1999. [Due to time constraints, the referenced paper

was not created. Instead, this RIAO 2000 paper was also used as the TREC-8 SDR overview in the TREC-8 Proceedings]

Graff, D., Wu, Z., MacIntyre, R., and Liberman, M., *The 1996 Broadcast News Speech and Language-Model Corpus*, Proc. DARPA Speech Recognition Workshop, February 1997.

Johnson, S.E., Jurlin, P., Moore, G.L., Spärck Jones, K., and Woodland, P.C., *The Cambridge University Spoken Document Retrieval System*, Proc ICASSP '99, Vol. 1, pp 49-52, March 1999)

Kantor, P., and Voorhees, E.M., *The TREC-5 Confusion Track: Comparing Retrieval Methods for Scanned Text*, Information Retrieval, In press – 2000.

Kubala, F., Colbath, S., Liu, D., Srivastava, A., Makhoul, J. *Integrated technologies for indexing spoken language*, Communications of the ACM, Volume 43, page 48, Feb. 2000.

Miller, D., Schwartz, R., Weischedel, R., Stone, R., *Named Entity Extraction from Broadcast News*, Proc. 1999 DARPA Broadcast News Workshop, March 1999.

Pallett, D., Fiscus, J., and Przybocki, M., *1996 Preliminary Broadcast News Benchmark Tests*, Proc. DARPA Speech Recognition Workshop, February 1997.

Pallett, D.S., Fiscus, J.G., Martin, A., Przybocki, M.A., *1997 Broadcast News Benchmark Test Results: English and Non-English*, Proc. DARPA Broadcast News Transcription and Understanding Workshop, February 1998.

Pallett, D.S., Fiscus, J.G., Garofolo, J.S., Martin, A., Przybocki, M., *1998 Broadcast News Benchmark Test Results: English and Non-English Word Error Rate Performance Measures*, Proc. DARPA Broadcast News Workshop, February 1999.

Przybocki, M.A., Fiscus, J.G., Garofolo, J.S., Pallett, D.S., *1998 Hub-4 Information Extraction Evaluation*, Proc. 1999 DARPA Broadcast News Workshop, March 1999.

Singhal, A., Pereira, F., *Document Expansion for Speech Retrieval*, Proc. SIGIR '99, 1999.

Voorhees, E., Garofolo, J., and Spärck Jones, K., *The TREC-6 Spoken Document Retrieval Track*, Proc. DARPA Speech Recognition Workshop, February 1997.

Voorhees, E., Garofolo, J., and Spärck Jones, K., *The TREC-6 Spoken Document Retrieval Track*, TREC-6 Notebook, Nov. 1997.

Voorhees, E.M., Harman, D., *Overview of the Seventh Text REtrieval Conference (TREC-7)*, Proc. TREC-7, November 1998.

Voorhees, E.M., Harman, D., *Overview of the Sixth Text REtrieval Conference (TREC-6)*, Information Processing and Management, Vol. 36, No. 1, pp 3-35, January 2000.

TREC-6 SDR Participant Publications (http://trec.nist.gov/pubs/trec6/t6_proceedings.html)

Abberley, D., Renals, S., *The THISL Spoken Document Retrieval System*, University of Sheffield, UK, G. Cook, T. Robinson, Proc. TREC-6, Nov. 1997.

Allan, J., Callan, J., Croft, W.B., Ballesteros, L., Byrd, D., Swan, R., Xu, J., *INQUERY Does Battle With TREC-6*, Proc. TREC-6, Nov. 1997.

Crestani, F., Sanderson, M., Theophylactou, M., Lalmas, M., *Short Queries, Natural Language and Spoken Document Retrieval: Experiments at Glasgow University*, Proc. TREC-6, Nov. 1997.

Fuller, M., Kaszkiel, M., Ng, C.L., Vines, P., Wilkinson, R., Zobel, J. *MDS TREC6 Report*, Proc. TREC-6, Nov. 1997.

Mateev, B., Munteanu, E., Sheridan, P., Wechsler, M., Schäuble, P., *ETH TREC-6: Routing, Chinese, Cross-Language and Spoken Document Retrieval*, Proc. TREC-6, Nov. 1997.

Oard, D.W., Hackett, P., *Document Translation for Cross-Language Text Retrieval at the University of Maryland*, Proc. TREC-6, Nov. 1997.

- Siegler, M.A., Slattery, S.T., Seymore, K., Jones, R.E., Hauptmann, A.G., Witbrock, M.J., *Experiments in Spoken Document Retrieval at CMU*, Proc. TREC-6, Nov. 1997.
- Singhal, A., Choi, J., Hindle, D., Pereira, F., *AT&T at TREC-6: SDR Track*, Proc. TREC-6, Nov. 1997.
- Smeaton, A.F., Quinn, G., Kelledy, F., *Ad hoc Retrieval Using Thresholds, WSTs for French Monolingual Retrieval, Document-at-a-Glance for High Precision and Triphone Windows for Spoken Documents*, Proc. TREC-6, Nov. 1997.
- Walker, S., Robertson, S.E., Boughanem, M., Jones, G.J.F., Spärck Jones, K., *Okapi at TREC-6 Automatic ad hoc, VLC, routing, filtering and QSDR*, Proc. TREC-6, Nov. 1997.

TREC-7 SDR Participant Publications (http://trec.nist.gov/pubs/trec7/t7_proceedings.html)

- Abberley, D., Renals, S., Cook, G., Robinson, T., *Retrieval Of Broadcast News Documents With the THISL System*, Proc. TREC-7, Nov. 1998.
- Allan, J., Callan, J., Sanderson, Xu, J., *INQUERY and TREC-7*, Proc. TREC-7, Nov. 1998.
- Dharanipragada, S., Franz, M., Roukos, S., *Audio-Indexing For Broadcast News* (reference to TREC-6 SDR), Proc. TREC-7, Nov. 1998.
- Ekkelenkamp, R., Kraaij, W., van Leeuwen, D., *TNO TREC7 site report: SDR and filtering*, Proc. TREC-7, Nov. 1998.
- Fuller, M., Kaszkiel, M., Ng, C., Wu, M., Zobel, J., Kim, D., Robertson, J., Wilkinson, R., *TREC 7 Ad Hoc, Speech, and Interactive tracks at MDS/CSIRO*, Proc. TREC-7, Nov. 1998.
- Henderson, G.D., Schone, P., Crystal, T.H., *Text Retrieval via Semantic Forests: TREC7*, Proc. TREC-7, Nov. 1998.
- Johnson, S.E., Jurlin, P., Moore, G.L., Spärck Jones, K., Woodland, P.C., *Spoken Document Retrieval for TREC-7*, Proc. TREC-7, Nov. 1998.
- Nowell, P., *Experiments in Spoken Document Retrieval at DERA-SRU*, Proc. TREC-7, Nov. 1998.
- Oard, D.W., *TREC-7 Experiments at the University of Maryland*, Proc. TREC-7, Nov. 1998.
- Siegler, M., Berger, A., Hauptmann, A., Witbrock, M., *Experiments in Spoken Document Retrieval at CMU*, Proc. TREC-7, Nov. 1998.
- Singhal, A., Choi, J., Hindle, D., Lewis, D.D., Pereira, F., *AT&T at TREC7*, Proc. TREC-7, Nov. 1998.

TREC-8 SDR Participant Publications (http://trec.nist.gov/pubs/trec8/t8_proceedings.html)

- Abberley, D., Ellis, D., Renals, S., Robinson, T., *The THISL SDR System At TREC-8*, Proc. TREC-8, Nov. 1999.
- Allan, J., Callan, J., Feng, F-F., Malin, D., *INQUERY and TREC-8*, Proc. TREC-8, Nov. 1999.
- Franz, M., McCarley, J.S., Ward, R.T., *Ad hoc, Cross-language and Spoken Document Information Retrieval at IBM*, Proc. TREC-8, Nov. 1999.
- Fuller, M., Kaszkiel, M., Kimberley, S., Ng, C., Wilkinson, R., Wu, M., Zobel, J., *The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8*, Proc. TREC-8, Nov. 1999.
- Gauvain, J-L., de Kercadio, Y., Lamel, L., Adda, G., *The LIMSI SDR System for TREC-8*, Proc. TREC-8, Nov. 1999.
- Han, B., Nagarajan, R., Srihari, R., Srikanth, M., *TREC-8 Experiments at SUNY Buffalo*, Proc. TREC-8, Nov. 1999.
- Kraaij, W., Pohlmann, R., Hiemstra, D., *Twenty-One at TREC-8: using Language Technology for Information Retrieval*, Proc. TREC-8, Nov. 1999.

- S.E. Johnson, P. Jourlin, K. Spark Jones, P.C. Woodland, *Spoken Document Retrieval for TREC-8 at Cambridge University*, Proc. TREC-8, Nov. 1999.
- Siegler, M., Jin, R., Hauptmann, A., *CMU Spoken Document Retrieval in TREC-8: Analysis of the role of Term Frequency TF*, Proc. TREC-8, Nov. 1999.
- Singhal, A., Abney, S., Bacchiani, M., Collins, M., Hindle, D., Pereira, F., *AT&T at TREC-8*, Proc. TREC-8, Nov. 1999.

Overview of the TREC-8 Web Track

David Hawking*

CSIRO Mathematical and Information Sciences,

Canberra, Australia

Ellen Voorhees

National Institute of Standards and Technology

Gaithersburg MD, USA

Nick Craswell and Peter Bailey

Department of Computer Science, ANU

Canberra, Australia

David.Hawking@cmis.csiro.au, Ellen.Voorhees@nist.gov, {nick,peterb}@cs.anu.edu.au

October 26, 2000

Abstract

The TREC-8 Web Track defined ad hoc retrieval tasks over the 100 gigabyte VLC2 collection (Large Web Task) and a selected 2 gigabyte subset known as WT2g (Small Web Task). Here, the guidelines and resources for both tasks are described and results presented and analysed.

Performance on the Small Web was strongly correlated with performance on the regular TREC Ad Hoc task. Little benefit was derived from the use of link-based methods, for standard TREC measures on the WT2g collection. The number of inter-server links within WT2g may have been too small or it may be that link-based methods would have worked better with different types of query and/or with different types of relevance judgment. In fact, a small number of link-based runs proved to be much more effective than their content-only baseline at finding documents which linked to documents judged relevant.

A variety of issues were investigated by participants in the Large Web Task. One group investigated the use of PageRank scores and found no benefit on standard TREC measures. Engineering improvements by several groups led to either considerable reduction in query processing time or reduction in the amount of hardware necessary to maintain comparable performance.

1 Introduction

The TREC-8 Web Track activities centred on two tasks: the Small and the Large Web Tasks. The latter featured the 100 gigabyte, 18.5 million webpage VLC2 collection described in last year's VLC Track overview [Hawking et al. 1998] and on the Web Track website [ACSys]. The former made use of a 2 gigabyte, 250,000 document subset of the VLC2, distributed on CD-ROM as the WT2g collection. Note that documents in WT2g are given different document numbers than the ones they had in the VLC2 (to enable easy extraction of the document) but include the original document numbers within DOCIDNO tags.

As it turned out, the Large and Small Web sub-tracks had very little in common apart from the use of spidered Web data. Accordingly, they will be described separately.

*The authors wish to acknowledge that this work was carried out partly within the Cooperative Research Centre for Advanced Computational Systems established under the Australian Government's Cooperative Research Centres Program.

2 Small Web Task

The focus of the Small Web Task was on answering two specific questions:

1. Do the best methods in the TREC Ad Hoc task also work best on the WT2g collection of Web data?
2. Can link information in Web data be used to obtain more effective search rankings than can be obtained using page content alone?

2.1 Topics and assessments

The Small Web Task used the TREC-8 Ad Hoc topics. Submissions were judged by NIST assessors using the same tools and presentation as used for the Ad Hoc documents. The small web and ad hoc documents for a given topic were judged by the same assessor, almost always the topic author. There was a single pool of documents for each topic, but since pools were sorted alphabetically by document number, all small web track documents appeared after the ad hoc documents. (Assessors were free to jump around in the pool when judging, but seldom did so.)

A public-domain HTML to ASCII converter, substantially modified by Ellen Voorhees, was used to render the entire web collection into ASCII. The rendered version is what the assessors judged. The rendering threw away all images, scripts, and frames, replacing them with a simple notice such as [IMAGE GOES HERE], unless the HTML provided ALT text in which case that was used (this occurred very rarely). The text of tables was retained, and a rough approximation of its formatting. Links were NOT rendered. This rendered collection is what was indexed to enable the assessor to do PRISE [NIST] searches during topic development. However, wholesale pre-editing of the source was needed to eliminate Word and PowerPoint documents, Chinese, Japanese, and other non-text data.

2.2 Judging pools

The number of runs judged was 27, giving a maximum pool size of 2700. The mean actual pool size was 950, 35.2 % of the maximum, while the mean number of relevant documents over the 50 topics is 45, which is 4.8 % of the number of documents judged.

41.2 % of the documents in the pool were contributed by both a content-only and a content-link run. 39 % of the pool was contributed by only content-only runs, and 19.8 % by only content-link runs. The statistics for the relevant documents are more skewed: 76.3 % of the relevant documents were found by runs of both types, 19.2 % of the relevant documents by only content-only runs, and 4.5 % of the relevant documents by only content-link runs.

Each of the 17 groups that submitted Small Web Task runs found some relevant documents that no other group retrieved in the top 100 ("unique relevants"). The largest totals over the 50 topics for unique relevants are 89 for Rutgers(Davison), 60 each for Claritech and RMIT, and 36 for IRIT. The totals for the other groups ranged from 5 to 30.

The pool statistics for the Small Web Task are roughly comparable to the statistics for the main Ad Hoc Task. For the Ad Hoc pools, the mean actual pool size was 1736 out of 7100 possible (24.5 %) and the mean number of relevant documents is 94 (5.4 % of what was judged). The number of unique relevants was comparatively larger, mostly reflecting the use of manual runs: the top three totals for unique relevants are 478 for MITI, 114 for Oracle, and 80 for IIT.

2.3 Definition of the WT2g dataset

In order to address the Small Web questions, a subset of the 100 gB VLC2 collection was needed which:

- was comparable in size to the TREC Ad Hoc collection (so as not to discourage participation, and to avoid perturbing collection parameters more than necessary);

- was likely to contain a reasonable quantity of material relevant to the TREC-8 Ad Hoc topics;
- included naturally defined sub-collections; and
- contained an interesting quantity of closed hyperlinks (with both source and target page within the subcollection).

The second and third requirements ruled out a uniform 2 % sample.

The method of choosing the WT2g subset collection was entirely heuristic. We started by identifying all the distinct hosts represented in the 100 gigabyte collection. Then we counted how many relevant documents were found in the VLC tasks (using TREC-7 ad hoc topics) and ranked the hosts in order of decreasing relevant document density. Finally, we collected all the documents from the top-ranked hosts until we reached a little over 2 gigabytes of data. The number of hosts represented is 956.

We expected that:

- the much higher density of relevant documents on TREC-7 topics than the average for the VLC2 would lead to a similarly higher density for TREC-8 topics; and
- because all available documents from each host were included, the proportion of dead links in this 2 gigabyte sample would be much less than for a randomly chosen sample of the same size.

Table 1: The density of known relevant documents in VLC2, WT2g and the current TREC Ad Hoc collections for TREC-7 and TREC-8 Ad Hoc topics. The original T7 judgments for WT2g were obtained from runs against the whole VLC2 collection and were understood to be incomplete. After TREC-7, ACSys judged some additional documents within WT2g for the T7 Ad Hoc topics. These are reported in the line marked "T7+new".

Judgments	Collection	Density of relevant docs
T7	VLC2	6482/18571671 = 0.03 %
T7	WT2g	3105/247491 = 1.25 %
T7+new	WT2g	6495/247491 = 2.62 %
T7	Ad Hoc	4674/528155 = 0.89 %
T8	WT2g	2279/247491 = 0.92 %
T8	Ad Hoc	4728/528155 = 0.90 %

Table 1 and Figure 1 show the densities of known relevant documents for the TREC-7 and TREC-8 Ad Hoc topics within various collections. Naturally, there may be considerable variation from one topic to another.

NIST assessors referred to the WT2g collection during the process of ad hoc topic generation. The assessors checked the number of relevant documents in the Web collection once they had a candidate topic from searching the ad hoc collection. The procedure was the same for both collections:

1. Use PRISE [NIST] to retrieve 25 documents.
2. If, after judging 25 documents there are at least 1 and less than 20 relevant documents, perform feedback and judge the resulting top 100; otherwise stop since candidate topic is not acceptable.

There was a nominal cut-off of at least 10 relevant documents in each collection to be selected as a final topic, but slightly less than that were accepted on some topics.

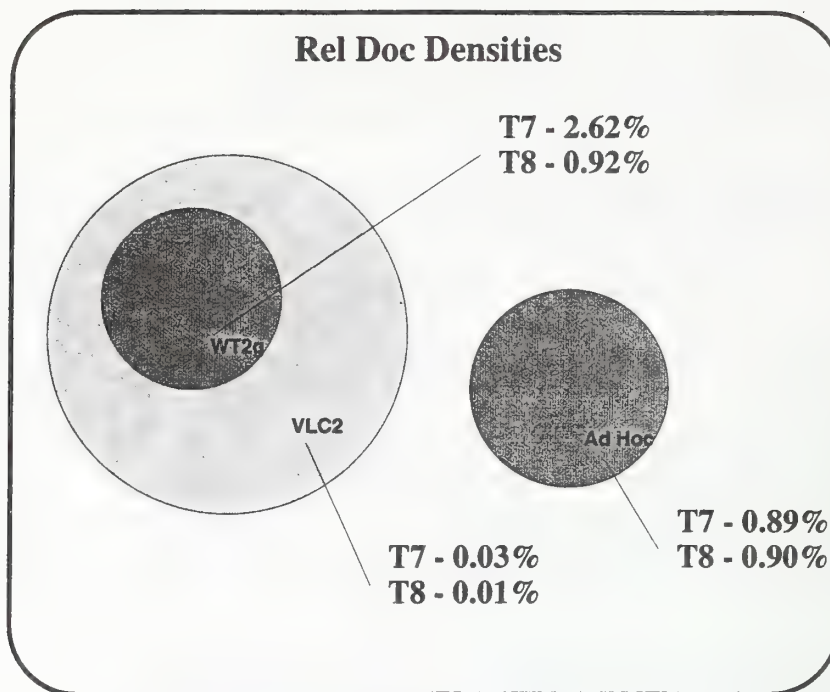


Figure 1: Pictorial representation of the data in Table 1.

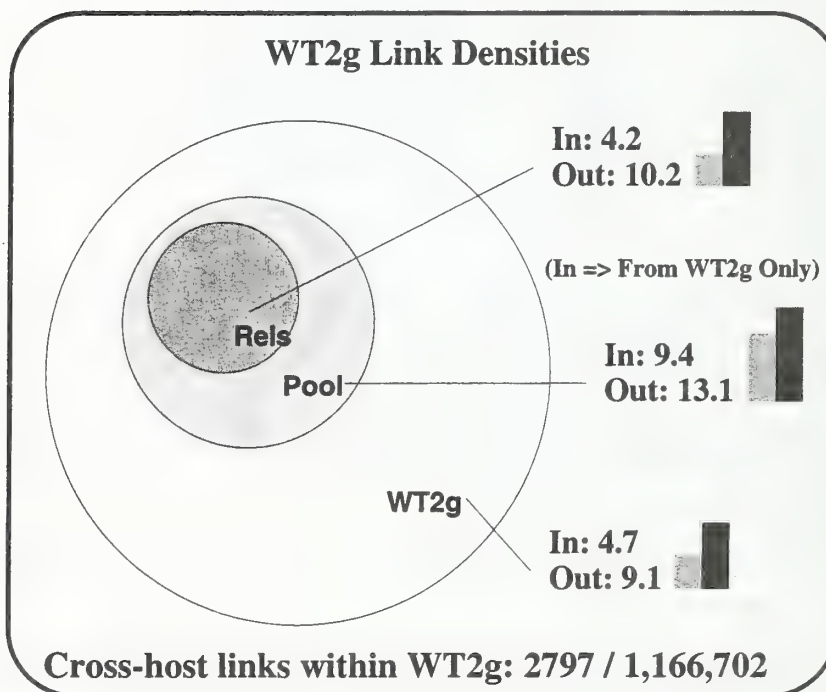


Figure 2: Pictorial representation of the data in Table 2.

2.4 Connectivity data

Nick Craswell developed software for extracting hyper-link connectivity information from WT2g. ACSys made that data available in two ways. First a connectivity server was made available on the Web. For each input URL the server would respond with a list of incoming links from other WT2g documents and outgoing links. However, because of network latencies and the extra client-side coding needed to resolve each URL to a canonical form and quote any special characters to avoid confusing the CGI script, the connectivity server was difficult to use. Accordingly, the connectivity data was also distributed by ftp in a highly compressed format based on WT2g document numbers. The out-links file consisted of, for each document d , the document numbers of the documents d links to. The in-links file was similar, but listed for each d the document numbers of documents linking to d .

2.5 Interconnectedness of WT2g

Table 2: The average number of outgoing links per document for various source sets, broken down by destination. The number of documents in WT2g is 247,491, the number in the assessment pool is 35,089 and the number in the relevant set is 2279.

Link source	Link target	Total Links	Links per source doc	Links per target doc
WT2g	Universe	2,259,952	9.13	-
WT2g	WT2g	1,166,702	4.71	4.71
WT2g	Assessment pool	330,295	1.33	9.41
WT2g	Relevant set	9512	0.04	4.17
Assessment pool	Universe	460,449	13.12	-
Assessment pool	WT2g	217,288	6.19	0.88
Assessment pool	Assessment Pool	88,468	2.52	2.52
Assessment pool	Relevant set	3579	0.10	1.57
Relevant set	Universe	23,337	10.24	-
Relevant set	WT2g	10,843	4.76	0.04
Relevant set	Assessment Pool	5181	2.27	0.15
Relevant set	Relevant set	711	0.31	0.31

Table 2 and Figure 2 show the density of links between different sets of documents. On average, each document within the collection includes 9.13 outgoing links. 52 % of these links reference another document within WT2g but only 0.12 % reference a different server within WT2g. It is not known at this stage, what proportion of the dead links (those whose target lies outside WT2g) are inter-server links and how many are references to same-server pages which happen to be missing from the VLC2¹.

2.6 Summary of participation

Seventeen groups submitted a total of 44 runs, 24 content-only and 20 making use of links.

2.7 Content-only runs

Groups which submitted exactly corresponding runs in the Ad Hoc and Small Web Tasks were asked to supply run identifiers and evaluation results for the Ad Hoc task. The corresponding average precision scores for these runs on the two tasks are tabulated in Table 4 and plotted against each other in Figure 3. It should be noted that some of the pairs of runs did not exactly correspond. AT&T used duplicate

¹Note that if any pages from a server are included in the WT2g, all VLC2 pages from that server are also included.

Table 3: The best performing content-only runs for each of the 17 participating groups, presented in order of decreasing average precision.

Group	Run tag	Ave. prec.	P@20
Microsoft	ok8wmx	0.3829	0.4520
Fujitsu	Flab8wtdnN	0.3405	0.4010
UMass	INQ620	0.3327	0.4130
MDS/RMIT	mds08w1	0.3220	0.3860
UNeuchatel	UniNEW2Ct	0.3150	0.3940
AT&T	att99wtde	0.3113	0.4110
UWaterloo	uwmt8w0	0.3066	0.3620
ACSys	acsys8wm	0.3009	0.3870
Claritech	CL99WebM	0.2889	0.2880
Illinois	iit99wt1	0.2265	0.3150
Dublin City Uni	DCU99C01	0.1936	0.2510
Seoul Uni	Scai8Web1	0.1854	0.2660
IRIT, Toulouse	Mer8Wctd	0.1638	0.2430
Rutgers	disco2	0.1023	0.1270
Oslo	hio1	0.0927	0.1420
UIowa	uiowaweb1	0.0747	0.1450
UNC	isw50t	0.0291	0.0830

elimination when controlling feedback on the Web runs but not in Ad Hoc. IRIT results constitute the most obvious outlier, but it is not yet clear why.

2.8 Exploitation of links

Table 5 summarises the methods used by the Small Web participants.

Tables 6 – 8 summarise the average precision, P@20 and total-relevant-documents-retrieved scores for each group which submitted at least one content-plus-link run. Each line in these tables gives the baseline performance in Column 2 and the corresponding performance for each link run in the remaining columns. Unfortunately, it is not completely clear that the only difference between the content-plus-link runs and the baseline is the use of links.

The differences between content-plus-link runs and the corresponding baseline are mostly very small and usually negative. The few large differences were all negative.

2.9 Duplicate elimination

Participants were not encouraged to apply duplicate elimination to their runs. It would thus be unfair to penalise runs which included duplicates within their rankings.

Despite a claim that there is at least one topic for which all the relevant documents are clones of each other, it is unlikely that the presence of duplicates would distort relative performance significantly. This is because of averaging over 50 topics and because the presence of irrelevant near-duplicates can degrade performance.

Future Web tracks may adopt evaluation measures which do not reward the presentation of multiple “near-duplicate” pages. However, the following issues need to be resolved:

- What constitutes a near-duplicate?

Table 4: Pairs of corresponding runs in Ad Hoc and Small Web Tasks.

Ad Hoc		Small Web	
Run tag	Ave. prec.	Run tag	Ave. prec.
Scai8Adhoc	.1461	Scai8Web1	.1854
acsys8amn	.2353	acsys8wm	.3009
ok8amxc	.3169	ok8wmx	.3829
att99atdc	.3089	att99wtde	.3091
att99atde	.3165	att99wtde	.3113
INQ603	.2659	INQ620	.3327
unofficial	.2293	mds08w1	.3220
Mer8Adtd1	.2231	MerWctd	.1638
uwmt8a0	.2143	uwmt8w0	.3066
isa50t	.027	isw50t	.029

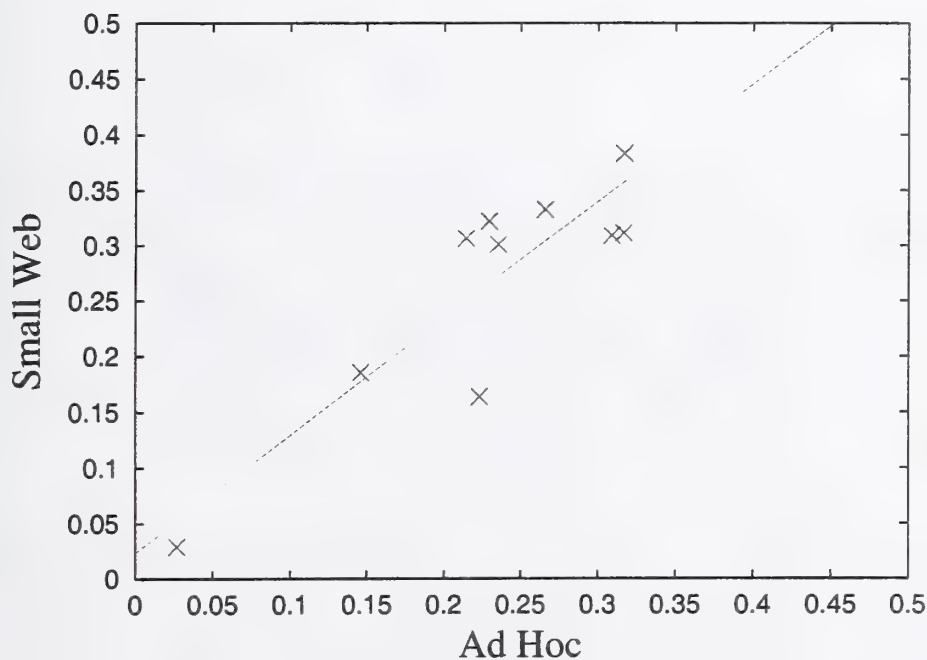


Figure 3: Average precision on the Small Web Task plotted against average precisions on the Ad Hoc task for pairs of runs believed to correspond closely, as per Table 4. Also shown is the line of best (least-squares) fit. The Pearson R coefficient of correlation is 0.884, which is significant at the 0.05 level (two-tailed).

Table 5: Link exploitation methods used by groups participating in the Small Web Task.

Group	Methods
MDS/RMIT	Sibling pages
UniNeuchatel	Kleinberg, PageRank, Spread. Act., PAS
ACSys	PageRank
IIT	Modified Kleinberg
DCU	Inlink/Outlink frequencies
Seoul Nat. Uni.	Score propagation along inlinks
IRIT	Spread. Act.
Rutgers	like Kleinberg
Oslo College	?
UIowa	?
Claritech	Kleinberg

Table 6: Comparison of average precision scores for runs using links with those of the corresponding baseline runs. Four link-based runs out of the 20 submitted achieved scores which were slightly higher (numerically) than their baselines. They are highlighted in boldface.

Group	baseline	links1	links2	links3
MDS/RMIT	0.3220	0.3047	0.2878	
UniNeuchatel	0.3150	0.3137		
UniNeuchatel	0.2739	0.2747		
ACSys	0.3009	0.3007	0.3007	0.2804
IIT	0.2265	0.2265	0.2264	
DCU	0.1936	0.1939	0.1921	
Seoul	0.1854	0.1819		
IRIT	0.1638	0.1488	0.1435	0.1401
Rutgers	0.1023	0.1087		
Oslo	0.0927	0.0972	0.0945	0.0859
UIowa	0.0747	0.0246		

Table 7: Comparison of P@20 scores for runs using links with those of the corresponding baseline runs. Four link-based runs out of the 20 submitted achieved scores which were slightly higher (numerically) than their baselines. They are highlighted in boldface.

Group	baseline	links1	links2	links3
MDS/RMIT	0.3860	0.3330	0.3590	
UniNeuchatel	0.3940	0.3940		
UniNeuchatel	0.3650	0.3690		
ACSys	0.3870	0.3870	0.3700	0.3870
IIT	0.3150	0.3150	0.3150	
DCU	0.2510	0.2490	0.2510	
Seoul	0.2660	0.2660		
IRIT	0.2430	0.1950	0.2160	0.2130
Rutgers	0.1270	0.1110		
Oslo	0.1420	0.1670	0.1580	0.1430
UIowa	0.1450	0.0290		

Table 8: Comparison of total relevant documents retrieved across all 50 topics, for runs using links with those of the corresponding baseline runs. The link-based runs which retrieved more relevant documents than their baselines are shown in bold.

Group	baseline	links1	links2	links3
MDS/RMIT	1872	1872	1878	
UniNeuchatel	1880	1869		
UniNeuchatel	1796	1795		
ACSys	1835	1834	1748	1834
IIT	1575	1572	1568	
DCU	1017	1017	1017	
Seoul	1500	1504		
IRIT	1286	1338	1258	1352
Rutgers	1041	1072		
Oslo	1292	1288	1394	1176
UIowa	1074	1074		

Table 9: Number of directly and indirectly relevant documents found in the WT2g collection. Note that documents may count more than once – once for each topic for which they are relevant.

Type	Number
Directly relevant	2279
Directly OR indirectly relevant	8838
Directly AND indirectly relevant	242
Indirectly but not directly relevant	6559

- How to score near-duplicates within a ranking. Zero for all near-duplicates after the first? Or, fractional scores?

2.10 Scoring taking into account links

It is possible that a link-based retrieval method may return a significant number of documents which, although they contain little or no relevant content, contain links to relevant documents. Such indirectly relevant documents are of value to a searcher in a Web search context because they provide a low-cost path to relevant documents.

The benefits of link-based retrieval may thus be underestimated because `trec_eval` does not take this into account. ACSys has attempted to determine whether this was the case by looking at the indirectly relevant documents retrieved by the runs listed in Table 7.

The WT2g connectivity data (see http://pastime.anu.edu.au/WAR/WT2g_Links/ilink_WTonly.gz and the Small Web qrels file were used to find the set of documents which link directly to relevant documents. Table 9 gives the numbers of directly and indirectly relevant documents.

Table 10: Additional relevant documents found when documents which directly link to relevant documents are considered to be indirectly relevant. The left part of the table considers documents found in the top 20 rankings and the right part considers documents found anywhere within the top 1000 results. In each part, the base column shows the total indirectly relevant documents found across all 50 topics. The number of indirectly relevant documents found by the link-based runs is shown relative to the number found by the corresponding baseline run. If every indirectly relevant document were considered to have the same weight as a directly relevant one, each indirectly relevant document found in the top 20 would add 0.001 to the original precision @ 20.

Group	top 20				top 1000			
	base	links1	links2	links3	base	links1	links2	links3
MDS/RMIT	17	+6	+3		525	0	+85	
UniNeuchatel	13	0			597	+838		
UniNeuchatel	13	0			590	+772		
ACSys	22	0	+1	0	549	-1	-53	-1
IIT	18	0	0		476	+344	+254	
DCU	12	0	0		137	0	0	
Seoul	16	+2			418	+37		
IRIT	17	+13	+10	+6	438	+96	-13	+17
Rutgers	11	+82			283	+239		
Oslo	32	+1	+2	-14	516	+16	+34	-62
UIowa	25	+17			394	0		

Table 10 reports the number of indirectly but not directly relevant documents included in the runs listed in Table 7, both in the full (top 1000) rankings and in the top 20 rankings. The results are presented so as to highlight any differential tendency of link-based runs to find indirectly relevant documents.

Considering the "top 1000" part of the table, several link-based runs show differentially higher retrieval of indirectly relevant documents. If all indirectly relevant documents are considered to be as valuable as directly relevant ones, the total set of relevant documents nearly quadruples in size, recall values for the top 1000 rankings decline sharply and the ordering of several (content, content+link) pairs changes. Most notable of these changes are those of the University of Neuchatel whose content+link runs out-recall their corresponding baselines by 33 % (2477 v. 3304) and 32 % (2386 v. 3157) and IIT whose content+link runs now out-recall the baseline by up to 17 %. These comparisons are made on the basis of total number of relevant documents retrieved over the 50 topics.

Inspection of the table suggests that link methods used by the University of Neuchatel, by IIT and in one of the IRIT runs resulted in differentially greater retrieval of indirectly relevant documents to an extent that might possibly change the ranking of corresponding pairs of runs on recall.

Considering the "top 20" part of Table 10, the Rutgers link-based run has a much higher differential retrieval of indirectly relevant documents than any other link-based run. If every indirectly relevant document were accorded the same weight as a directly relevant one, the Rutgers baseline P@20 would increase to 0.138 (from 0.127) and the link-based run to 0.204 (from 0.110). This appears to be the only pair of runs for which the consideration of indirectly relevant pages may change the ranking of the runs on P@20. Furthermore, the benefit implied by these figures is almost certainly overstated, due to the assumption of equal worth for directly and indirectly relevant pages.

The worth of an indirectly relevant document to a searcher depends upon how easy it is to find the link to the directly relevant page(s). This is influenced by page layout factors not easily determinable automatically. For example:

1. whether the visual rendition of the link attracts attention;
2. whether the link is at the top of the document or in some other prominent position;
3. whether the anchor and context of the link allow the searcher to identify that the target of the link is likely to be relevant;
4. whether there are other similarly attractive links which, in fact, lead to irrelevant pages.

On average, the value of an indirectly relevant page is likely to be considerably less than that of a directly relevant page. Accordingly, scores on TREC measures, revised to take into account indirect relevance, have not been presented because they would depend upon an arbitrary assignment of relative weight for indirectly relevant pages.

2.11 Small Web Task discussion and conclusions

The University of Neuchatel and Fujitsu Laboratories report that they could find no correlation between relevance on the TREC-7 topics and link-based measures.

It seems fairly clear that, in this year's Small Web Task, no measurable benefit was gained on standard TREC retrieval measures through use of links. A small number of link-based runs benefited substantially on recall, and one on P@20 provided that indirectly relevant documents are assigned the same value as directly relevant ones.

The following questions arise:

1. Is the WT2g collection big enough and does it include enough links to permit effective operation of the link-based methods? Figure 2 shows that there are in fact a lot of links, but only a very small number of cross-server links.
2. Are link-based methods more likely to be effective for types of information need other than those modelled by TREC Ad Hoc topics. For example, locating a library's on-line catalogue or the home page of a particular travel agent.
3. Would link-based methods seem more effective if the TREC relevance assessment model were expanded to recognise that some pages are much more valuable than those which are merely relevant. For example, the desired on-line catalogue page may be of far more use to the searcher than learned papers about library cataloguing systems.

3 Large Web Task

The Large Web Task was by no means as tightly focused as the Small Web Task. A number of different objectives were pursued by the individual participants. They are summarised as follows:

ACSys Investigate the use of link-based measures on the full VLC2 set. Reduce the hardware required for VLC2 processing as much as possible, even to the level of a mid-range laptop. Further study efficiency-effectiveness tradeoffs.

AT&T Test locally distributed IR based on content only.

Fujitsu Labs Comparison of BooleanConjunction+Ranked with Ranked. Efficiency issues. How can VLC2 be indexed using a single index? If multiple concurrent processes on a single processor are used to process queries, what is the optimum degree of parallelism?

Microsoft - Okapi Determine the effect on speed of stop list size, output size limitation, and use of memory vs. use of temporary files.

CityU/Microsoft - Pliers To demonstrate good query processing time and scale-up on a large cluster of machines.

UMass Determine whether UMass conventional retrieval techniques would be effective in the domain of web pages.

UNC Investigate the possibility of having very fast retrieval from a very large information space using a variant of Latent Semantic Indexing.

UWaterloo Fast automatic retrieval on natural language queries. Develop the *cover density ranking* method, using probability based reasoning. Experiment with variations on query length and use of plural/nonplural words in queries.

3.1 Large Web Task: Topics and assessments

ACSys obtained 100,000 "natural language" queries from both Alta Vista [AltaVista Company] and the Electric Monk [Electric Knowledge LLC]. These were censored by a `perl` script to remove possibly offensive queries (or queries which might produce offensive answers)² and random selections were made from the remainder until 10,000 queries were selected. These queries were numbered 20001-30000 and distributed to participants.

Participants were required to process all 10000 queries and to submit top 20 rankings to ACSys for judging. After submissions were received, a `perl` script was used to repeat the following until 60 topics had been accepted:

1. Randomly select a topic within the 20001 - 30000 range.
2. If the selected topic had fewer than 2 non-stopwords, it was eliminated. The stopwords list had 51 entries.
3. If there were at least two non-stopwords, the topic was presented to one of the judges for acceptance or rejection. She was asked to accept a topic if she felt she understood what the person who originally posed the query wanted and if she felt able to judge the relevance or otherwise of documents on that topic.

²As became obvious during the ACSys demonstration in the Web Track session at the conference, the list of 110 words to be censored was still missing a few entries!

22539 who are the current supreme court justices?
 24127 how to make a battery
 28771 where can i find the saints and the catholic church?
 24111 how to quit smoking?
 22905 how to write bibliographies
 22719 where can i find information on herbs?
 24698 what are the causes of runoff pollution
 26776 armstrong louis
 21826 where can i find information on the bahamas
 24183 how do volcanoes erupt
 28150 animal rights
 29001 where can i find information about the death penalty?
 22674 slobadan milosevic
 25597 how do rocks form?
 21475 how does a digital camera work?
 26981 where can i find information about the civil war
 24976 show me a list of vegetarian restaurants in new york city.
 22610 thalidomide and multiple sclerosis
 25060 old japanese science fictions movies
 26274 sinus infection
 27375 how do you play chess
 29906 where can i find information on the amazon river?
 20732 tell me about prozac
 26417 how do solar panels work?
 24816 hindenburg disaster
 28346 find information about american anarchists
 26533 why do feet smell?
 28850 what are the current ethnic conflicts in azerbaijan?
 25358 what are some psychological principles and attitudes for advertising
 28273 how to start business
 28854 what are the current ethnic conflicts in belarus?
 26817 where can i find statistics for education in the united states?
 27092 methodist sermons
 24790 where can i find information about the politic situation in israel
 23274 human genome project
 21055 how do i create a web site?
 28634 reasons for studying marketing
 20784 blood pressure
 25233 where can i find information on school violence?
 21247 where can i find information on russia?
 28677 where can i watch tv on the internet?
 21185 where can i find the best jokes?
 25663 how are hospitals prepared for y2k?
 28798 where can i find information about teenage alcohol abuse in the uk
 28846 teen alcohol abuse statistics for the uk
 27358 egyptian history

Figure 4: A sample of the judged queries used in the Large Web task. Note the two very similar "teen alcohol" queries at the bottom of the list. Note also the retention of probable query errors: "slobadan", "science fictions", and "politic situation".

Something less than 100 selections were eliminated and 718 were rejected by the judge. Of the 60 accepted, it transpired that our judge had accidentally accepted one, 27188 where can i find black escorts? which she really wanted to reject and she also accepted two topics likely to have the same answers: 24111 how to quit smoking? / 23728 how do i quit smoking? Topics 27188 and 23728 were rejected.

Sample accepted queries are shown in Figure 4. Note that two similar queries related to "teen alcohol" were both accepted.

The number judged dropped by a further one when one of the judges was unavailable for a few days after all other topics were finished. The resulting 57 topics were pruned to 50 by arbitrarily eliminating all the topics for which there were fewer than 5 relevant documents.

The pooled documents for each topic were presented to the assessors in order of increasing document length using the RAT (Relevance Assessment Tool) used in previous VLC track experiments. This time however, a text-only web browser [Lynx] was used to display documents in a way which rendered references and tables in a reasonable way (minus images).

The six assessors were all University graduates from specialties other than Computer Science or Librarianship. Three of them had served as VLC track judges in previous years.

3.2 Large Web Task: Efficiency-effectiveness results

The tradeoffs between efficiency and effectiveness are actually tradeoffs among five dimensions:

1. Speed of indexing;
2. Size of indexes;
3. Speed of query processing;
4. Query processing effectiveness; and
5. Cost.

Table 11 shows how the different runs submitted to the 100 gigabyte collection web track made these tradeoffs.

For each run submitted against the full 18.5 million document collection, Figures 5 and 6 show a 5-axis Kiviat diagram summarising performance on each of these dimensions. On each axis, best performance is represented by a point on the circumference. For effectiveness, best performance corresponds to maximum P@20 score whereas in each other case best performance corresponds to minimum score.

To illustrate the scaling process, the smallest index size was achieved by Fujitsu at 3.9 gigabytes. This minimum was divided by the actual index size for each run to give a scaled score of 1 for Fujitsu and a score of 0.1 for a hypothetical index of 39 gigabytes. Scaled scores of less than 0.05 are shown as 0.05 to prevent the creation of spikes which are too narrow to see.

Use of linear scaling in the Kiviat diagrams tends to exaggerate the differences between runs, whereas log scaling would have tended to homogenize them. The shape of the diagram indicates the degree to which that run achieved good performance (relative to the group) on one (or a couple of) dimensions at the expense of the others, or alternatively achieved a good balance between them. Good balance is indicated by a filled-out shape, best illustrated by the hypothetical "uniformly best" system shown at the top left of Figure 5.

The Kiviat diagrams shown in Figure 5 are considerably distorted by the inclusion of the UNC runs which achieved enormous query processing speed but very low precision. The diagrams in the results section of the TREC-8 proceedings are quite different because they do not include the UNC runs (which were submitted after the deadline.)

Table 11: Summary of Results for all submitted runs over the full 100 gigabyte collection. Note that the UNC runs were submitted after the deadline and include many unjudged documents. Note also that the query processing times reported for Fujitsu correspond to the case where queries were processed as a sequential batch. Fujitsu achieved better times on the same hardware using two processes: 0.40 seconds (f18wlnsb); 0.75 seconds (f18wlnsr); 0.39 seconds (f18wlsb). See the Fujitsu paper for details.

Group	Runid	Cost (k\$(US))	idx_time (hr.)	idx_size (gB)	qp_time (sec.)	p20
ACSys	acsys8lw0	7	8.48	5.78	3.74	0.3360
ACSys	acsys8lw0_pr1	7	104	6.46	3.91	0.3360
ACSys	acsys8lw0_pr10	7	104	6.46	3.87	0.3350
AT&T	att99vlci	115	8.62	23.9	0.516	0.55650
AT&T	att99vlcm	115	8.62	23.9	0.516	0.5470
Fujitsu	f18wlnsb	41	504	5.10	0.75	0.5100
Fujitsu	f18wlnsr	41	504	5.10	1.16	0.5080
Fujitsu	f18wlsb	41	504	3.95	0.54	0.5070
Microsoft	ok8v1	16	131.1	66.5	6.73	0.5280
Microsoft	ok8v2	16	131.1	66.5	5.35	0.5380
Microsoft/CityU	plt8wt1	82	3.04	10.6	1.62	0.5610
UMass	INQ650	215	268	53.8	39	0.5000
UNC	iswqd1	200	40	22	0.005	0.0000
UNC	iswqd2	200	40	22	0.005	0.0000
UWaterloo	uwmt8lw0	5	8.53	32	0.841	0.5720
UWaterloo	uwmt8lw1	5	8.53	32	0.735	0.5580
UWaterloo	uwmt8lw2	5	8.53	32	1.010	0.5650

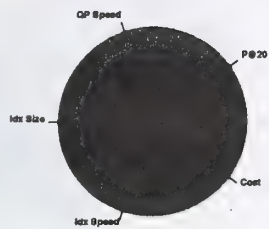
Table 12: Scale-up factors for CityU/Microsoft and UNC runs for BASE1:BASE10:VLC2.

Group	Measure	Scaleup 1:10	Scaleup 10:100	Scaleup 1:100
CityU/Microsoft	Index_build time	11.6	10.5	122
	Index size	8.23	8.76	72.1
	Query Proc. time	4.32	13.4	57.9
	P@20	1.62	1.29	2.08
UNC	Index_build time			57.1
	Index size			3.67
	Query Proc. time			0.83
	P@20			?

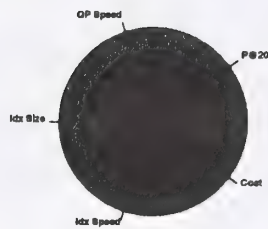


Large Web Runs - Linear Scaling - Sheet 1

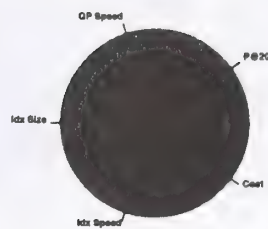
Figure 5: Composite results for all runs submitted in the Large Web Task. Note that the UNC runs were submitted after the deadline and consequently included a very high percentage of unjudged documents. Accordingly, their precision result is very low. However, their query processing was two orders of magnitude faster than the next best fastest, scaling other speed results into oblivion. The AT&T run was also unjudged due to a formatting problem. The All-Round Best is a hypothetical composition of the best-achieved result on each dimension. Finally, because ACSys co-ordinated the track, employed assessors and tabulated results, ACSys results should be regarded as unofficial. (Continued in Figure 6.)



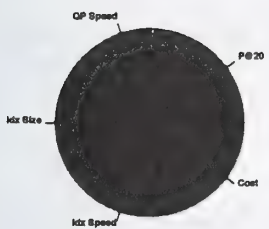
Microsoft - ok8v1



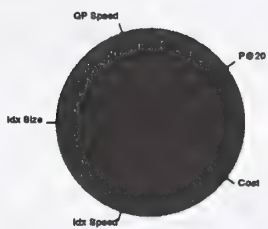
Microsoft - ok8v2



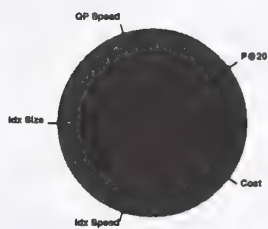
MS/City U - plt8wt1



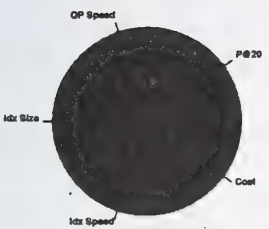
UMass - INQ650



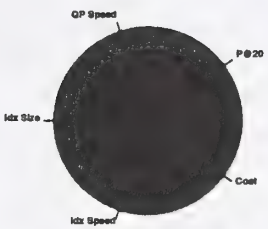
UNC - lswqd1



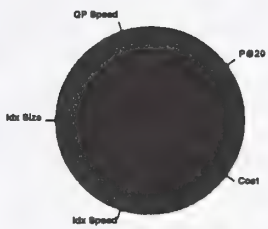
UNC - lswqd2



UWaterloo - uwmt8lw0



UWaterloo - uwmt8lw1



UWaterloo - uwmt8lw2

Large Web Runs - Linear Scaling - Sheet 2

Figure 6: Continuation of Figure 5.

3.3 Large Web Task: Scalability results

Two groups submitted scalability runs in which the VLC performance figures were compared with those of the BASE1 and BASE10 uniform samples. The scaleup factors for these runs are presented in Table 12.

3.4 Large Web Task: Exploiting links

ACSys (in the person of Nick Craswell) computed PageRank scores for all the documents in the VLC2. Results are reported in the ACSys paper in these proceedings.

In essence, computation of PageRanks took much longer than indexing but use of PageRanks increased query processing time only slightly (by an average of 0.15 seconds per query, less than 5 %.) However, the benefit in terms of query processing effectiveness was found to be negligible.

3.5 Large Web Task: Hardware resources

Several groups were successful in reducing the scale of hardware required to process the full VLC2 collection, compared to what they used in the TREC-7 VLC track. UWaterloo reduced their machinery from four PCs to two. ACSys used one PC instead of eight DEC Alphas and demonstrated query processing over the full collection on a Dell laptop using only the internal disk drives.

Fujitsu (who did not participate in TREC-7 VLC) demonstrated that, by eliminating non-English documents and HTML tags, the whole of the VLC2 could be represented in a single index of only 3.9 gigabytes.

3.6 Large Web Task: Other issues

The various other questions addressed by participants are covered in their own papers.

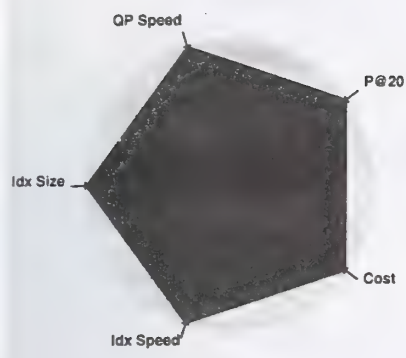
Acknowledgements

We are very much indebted to Brewster Kahle of the Internet Archive for making available the spidered data from which the VLC2 and WT2g collections are derived and to Alta Vista (Monika Henzinger and Michael Moricz) and the Electric Monk (Edwin Cooper) for providing large samples of queries from their logs. Thanks also to John O'Callaghan and Darrell Williamson (successive CEOs of ACSys) and Peter Langford (ACSys Centre Manager) for supporting the track.

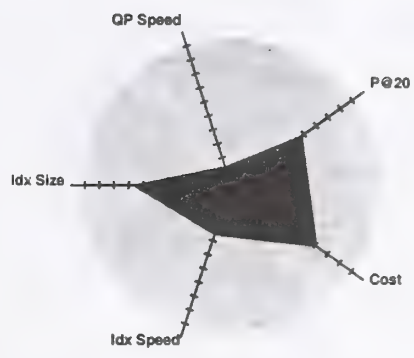
Finally, thanks are due to Sonya Welykyj, Penny Craswell, Clare Dyson, Zoe McKenzie, Greg Evans and Nick Clarke for their work in assessing Large Web submissions.

Bibliography

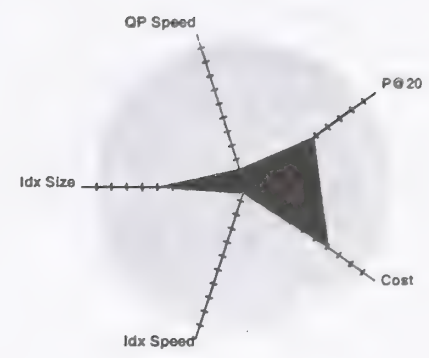
- ACSYS. TREC Web Tracks home page. <http://pastime.anu.edu.au/TAR/WT>.
- ALTAVISTA COMPANY. Alta Vista web page. <http://www.altavista.com/>.
- ELECTRIC KNOWLEDGE LLC. Electric Monk home page. <http://electricmonk.com/>.
- HAWKING, D., CRASWELL, N., AND THISTLEWAITE, P. 1998. Overview of TREC-7 Very Large Collection Track. In E. M. VOORHEES AND D. K. HARMAN Eds., *Proceedings of TREC-7* (Gaithersburg MD, November 1998), pp. 91-104. http://trec.nist.gov/pubs/trec7/t7_proceedings.html.
- LYNX. Lynx browser home page. <http://lynx.browser.org>.
- NIST. Guide to Z39.50/PRISE 2.0. <http://www.itl.nist.gov/div894/894.02/works/papers/zp2/zp2.html>.



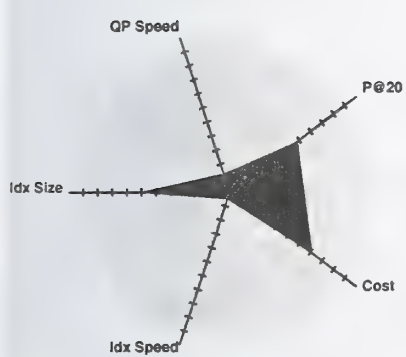
All-Round Best



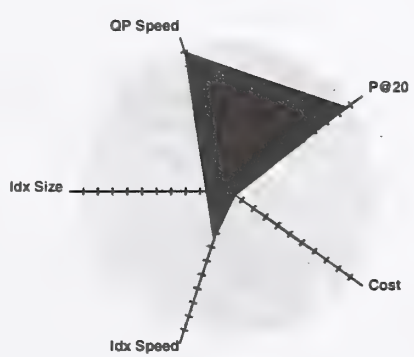
ACSys - acsys8lw0



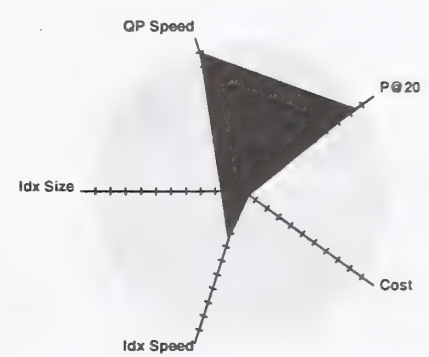
ACSys - acsys8lw0_pr1



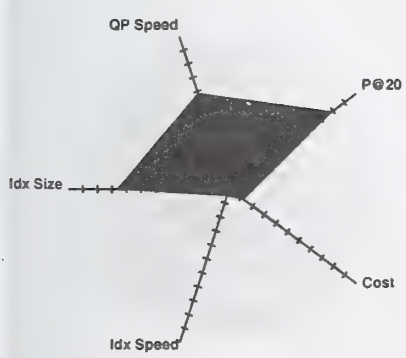
ACSys - acsys8lw0_pr10



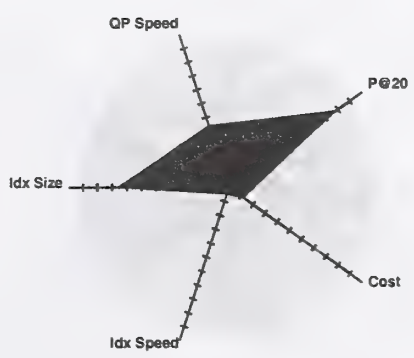
AT&T - att99vlci



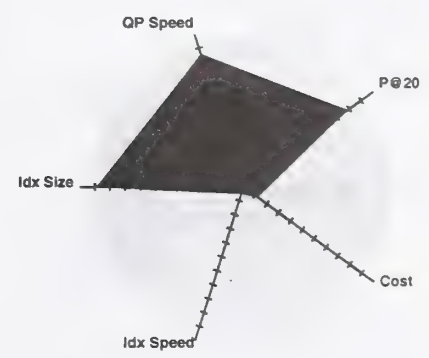
AT&T - att99vlcm



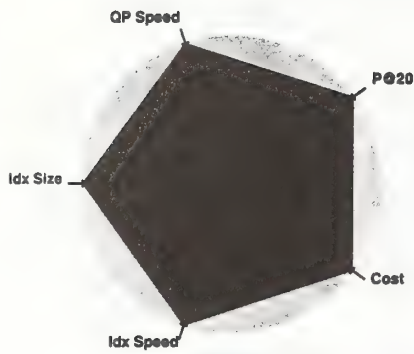
Fujitsu - fl8wlmsb



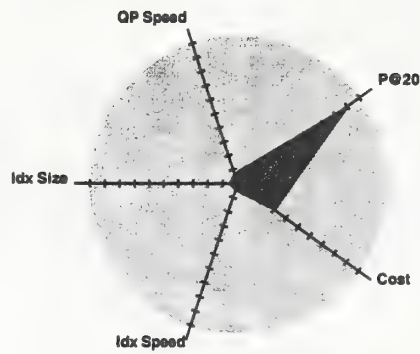
Fujitsu - fl8wlmsr



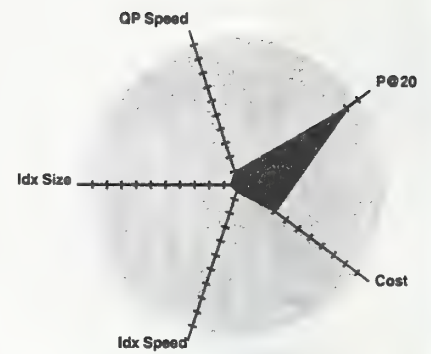
Fujitsu - fl8wlslb



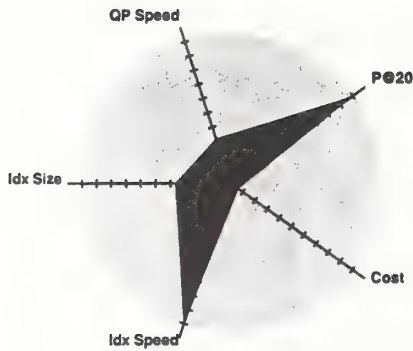
All-Round Best



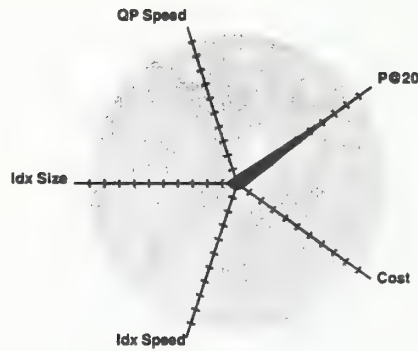
Microsoft - ok8v1



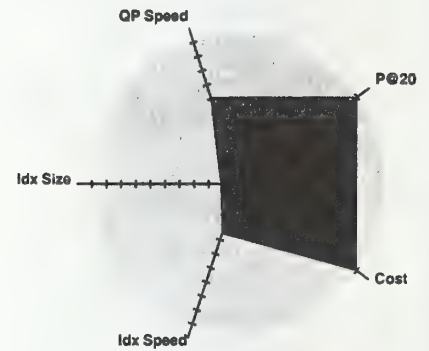
Microsoft - ok8v2



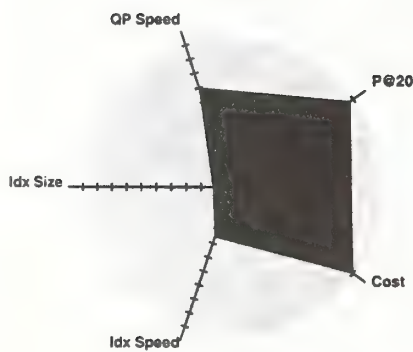
MS/City U - plt8wt1



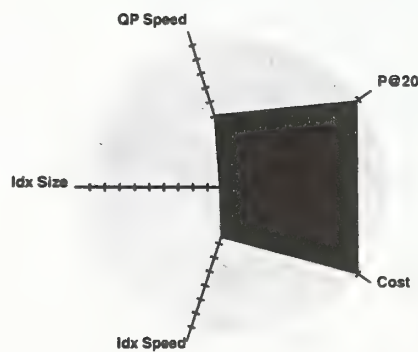
UMass - INQ650



UWaterloo - uwmt8lw0



UWaterloo - uwmt8lw1



UWaterloo - uwmt8lw2

Okapi/Keenbow at TREC-8

S E Robertson*

S Walker†

1 Summary

Automatic ad hoc and web track

Three ad hoc runs were submitted: long (title, description and narrative), medium (title and description) and short (title only). “Blind” expansion was used for all runs. The queries from the medium ad hoc run were reused for the small web track submission. Most of the negative expressions were removed from the narrative field of the topic statements, and a new expansion term selection procedure was tried.

Adaptive filtering

Methods were similar to those we used in TREC-7. Six runs were submitted.

VLC track

Two unexpanded ad hoc runs were submitted.

2 Okapi at TRECs 1-7

The Okapi search systems used for TREC are descendants of the Okapi systems developed at the Polytechnic of Central London¹ between 1982 and 1988 under a number of grants from the British Library Research & Development Department and elsewhere. These early Okapi systems were experimental highly-interactive reference retrieval systems of a probabilistic type, some of which featured automatic query expansion [1, 2, 3].

All the Okapi work in connection with TRECs 1-6 was done at the Department of Information Science, City University, London. Most of the Okapi TREC-7 entries were done from Microsoft Research, Cambridge (UK).

For TREC-1 [4], the low-level search functions were generalized and split off into a separate library — the Okapi Basic Search System (BSS). User interfaces or batch processing scripts access the BSS using a simple command language-like protocol. However, our TREC-1 results were very poor [4], because the classical Robertson/Sparck Jones weighting model [5] which Okapi systems had always used took no account of document length or within-document term frequency.

During TREC-2 and TREC-3 a considerable number of new term weighting and combination functions were tried; a runtime passage determination and searching package was added to the BSS; and methods of selecting good terms for routing queries were developed [6, 7]. During the TREC-2 work “blind” query expansion (feedback using terms from the top few documents retrieved in a pilot search) was tried for the first time in automatic ad hoc experiments, although we didn’t use it in the official runs until TREC-3. Our TREC-3 automatic routing and ad hoc results were good.

TREC-4 [8] did not see any major developments. Routing term selection methods were further improved.

By TREC-5 many participants were using blind expansion in ad hoc, several of them more successfully than Okapi [9, 10]. In the routing, we tried to optimize term weights after selecting good terms (as did at least one other participant); our routing results were again among the best, as were batch filtering runs.

In TREC-6 [11] we continued to investigate blind expansion, with mixed results. We also introduced a new weighting function designed to make use of documents known or assumed to be non-relevant. In routing and filtering

*Microsoft Research Ltd, 1 Guildhall Street, Cambridge CB2 3NH, UK, and City University, London, UK. email ser@microsoft.com

†Microsoft Research Ltd, 1 Guildhall Street, Cambridge CB2 3NH, UK. email sw@microsoft.com

¹Now the University of Westminster.

we continued to extend the optimization procedure, including a version of simulated annealing. Again our routing and filtering results were among the best. The Okapi BSS was modified to handle large databases for the VLC track.

We entered the adaptive filtering track, with fairly good results, for the first time in TREC-7 ([12]). Routing and batch filtering were dropped.

3 The system

At the Microsoft Research laboratory in Cambridge, we are developing an evaluation environment for a wide range of information retrieval experiments. This environment is called Keenbow. The Okapi BSS is now seen as a component of Keenbow.

3.1 The Okapi Basic Search System (BSS)

The BSS, which has been used in all Okapi TREC experiments, is a set-oriented ranked output system designed primarily for probabilistic-type retrieval of textual material using inverted indexes. There is a family of built-in weighting functions as defined below (equation 1) and described more fully in [7, Section 3]. In addition to weighting and ranking facilities it has the usual boolean and quasi-boolean (positional) operations and a number of non-standard set operations. Indexes are of a fairly conventional inverted type. There have been no major changes to the BSS during TREC-8.

Weighting functions

All TREC-8 searches used varieties of the Okapi **BM25** function first used in TREC-3 (equation 1).

$$\sum_{T \in Q} w^{(1)} \frac{(k_1 + 1)tf}{K + tf} \frac{(k_3 + 1)qtf}{k_3 + qtf} \quad (1)$$

where

Q is a query, containing terms T

$w^{(1)}$ is the Robertson/Sparck Jones weight [5] of T in Q

$$\log \frac{(r + 0.5)/(R - r + 0.5)}{(n - r + 0.5)/(N - n - R + r + 0.5)} \quad (2)$$

N is the number of items (documents) in the collection

n is the number of documents containing the term

R is the number of documents known to be relevant to a specific topic

r is the number of relevant documents containing the term

K is $k_1((1 - b) + b \cdot dl/avdl)$

k_1 , b and k_3 are parameters which depend on the nature of the queries and possibly on the database; k_1 and b default to 1.2 and 0.75 respectively, but smaller values of b are sometimes advantageous; in long queries k_3 is often set to 7 or 1000 (effectively infinite)

tf is the frequency of occurrence of the term within a specific document

qtf is the frequency of the term within the topic from which Q was derived

dl and $avdl$ are respectively the document length and average document length measured in some suitable unit.

Term ranking for selection

Prior to TREC-8 the method used was that proposed in [13] by which terms are ranked in decreasing order of

$$TSV = r \cdot w^{(1)} \quad (3)$$

This time a new method was tried; this is discussed in section 4.

Passage determination and searching

Since TREC-3 the BSS has had facilities for search-time identification and weighting of any sub-document consisting of an integral number of consecutive paragraphs. It was described, and some results reported, in [7]. Passage searching almost always increases average precision, by anything from 2%–10%, as well as recall and precision at the higher cutoffs. It often, perhaps surprisingly, reduces precision at small cutoffs, so is not used in pilot searches for expansion runs.

3.2 Hardware

All the TREC-8 processing was done at Microsoft Research, Cambridge. Most of the work was done on a 300 MHz Sun Ultra 10 with 256 MB and a Dell with two 400 MHz Pentium processors and 512 MB. Both machines were running Solaris 2.6. Mainly to cater for the VLC track there was about 170GB of disk storage, most of which was attached to the Sun. The network was 100MHz ethernet.

3.3 Database and topic processing

Text processing

For interactive purposes it is necessary to provide for the readable display of documents. Since we have not (yet) implemented a runtime display routine, nor adequate parsing and indexing facilities, for SGML data, all the TREC input text is subjected to batch conversion into a uniform displayable format before further processing. This is done by means of hacked up shell scripts specific to the input dataset. For most of the TREC data, output records have three fields: document number, any content unsuitable for indexing (or not to be searched—such as controlled descriptors in some datasets), and the searchable “TEXT” and similar portions. However, only two fields were used for the VLC98 collection.

Indexing

All the TREC text indexing was of the keyword type. A few multiword phrases such as “New York”, “friendly fire”, “vitamin E” were predefined and there was a pre-indexing facility for the conflation of groups of closely related or synonymous terms like “operations research” and “operational research” or “CIA” and “Central Intelligence Agency”. A stemming procedure was applied, modified from [14] and with additional British/American spelling conflation. The stoplist contained about 220 words.

Topic processing

Apart from the narrative field, which is only used in “long” topic queries, topic text is processed in the same way as text to be indexed. In the narrative field terms such as “document”, “describe(s)”, “relevan...”, “cite...” are deleted (in addition to the normal stop terms). There is a facility, new for TREC-8, for removing most negative expressions. This was tried on most of the past TRECs’ ad hoc data, and often made little difference. However, it was rarely detrimental, and appears beneficial in TREC-8 (see Table 2).

4 Term selection for query expansion

Readers who took part in early TRECs will recall discussions on the issue of “selective versus massive” query expansion. Many participants did some form of query expansion, particularly by extracting terms from previously known relevant documents in the routing task. The point at issue was whether one should include all candidate terms (possibly with small weights), or be selective and add only a small number. The Okapi team had a bias towards being very selective in expansion.

Since then, of course, many teams, including us, have applied and adapted the relevance feedback methods to blind expansion (pseudo-relevance feedback). The same problem arises; in fact in this case, there is an earlier stage of selection, namely the selection of a number of *documents* from the top of the initial retrieval ranking.

Our general method for query expansion following relevance feedback is to rank terms according to some measure of their likely contribution to the effectiveness of the search, and then to take a certain (fixed) number from the top of this ranking. We have used various measures for this ranking purpose, but a common feature of all such measures is that they are intended *only* for ranking, that is for measuring *relative* contribution; none is susceptible to the use of

an absolute threshold or criterion for inclusion. Last year's work on thresholding for filtering inspired us to consider the possibility of devising an absolute measure which could be compared to an absolute threshold. The potential advantages of such a measure are:

- If there are more good terms, more can be included, and vice-versa.
- The measure would take account of the number of relevant documents; one might expect to get stronger evidence for more good terms from more relevant documents.

The method discussed below is a first attempt at such a measure. We have applied it to the blind expansion problem, although it is in principle applicable to real relevance feedback as well. We have not yet tackled the additional selection problem for blind feedback (document selection), and therefore the second reason above does not really apply; however, there may be a secondary effect here, that if the initial retrieval generates a coherent set of documents, there is more chance of more good terms emerging from the analysis than if the initial search is poor and generates a more random set.

It is also worth pointing out that although the idea was inspired by the filtering-threshold problem, the method itself is very different.

4.1 Statistical significance of new terms

We are looking for new terms which are sufficiently strongly associated with relevance to contribute to performance. A measure which might satisfy the above requirement would be a measure of the statistical significance of any given term's association with relevance. Significance measures are typically on a scale which allows a comparison with an absolute likelihood of the null hypothesis (that is, an absolute probability of the observation given a hypothesis of no association). This absolute value might be (say) 5%, 1% or 0.1%; although the choice of level is largely arbitrary, any fixed single choice would satisfy the requirements given above, when applied to different topics or different numbers of relevant documents. For reasons given below, these specific values are not themselves appropriate to this case, but the principle remains.

As in most of our various term selection measures, we look primarily at term presence/absence (we have had very little success with methods which take account of term frequency). Thus the relation between the term and relevance (known or assumed) is defined by the usual 2×2 table:

	Relevant	Not relevant	
Term t present	r_t	$n_t - r_t$	n_t
Term not present	$R - r_t$	$N - R - n_t + r_t$	$N - n_t$
	R	$N - R$	N

(as usual, all documents not known or assumed to be relevant are assumed to be not relevant).

The null hypothesis is that the term is not associated with relevance. The likelihood of the null hypothesis, given the above data, may be approximated as follows. The probability of the term occurring in a document taken at random is n_t/N . We assume that the term occurs in a small proportion of the collection as a whole (for all terms of interest, this will be the case). Then the probability that it occurs in exactly r_t out of the R relevants is approximately

$$\left(\frac{n_t}{N}\right)^{r_t} \left(\frac{R}{r_t}\right) \quad (4)$$

The second factor is the number of ways we can choose r_t from R , and can be calculated as $\frac{R!}{r_t!(R-r_t)!}$.

4.2 The criterion

As indicated, a usual criterion for rejection of a null hypothesis would be a likelihood of less than (say) 1%. However, if we were to apply such a criterion in this case, we would be likely to get a great deal of noise. The reason is that there are so many terms in a text retrieval system to start with. Suppose that the total vocabulary in the system (indexed terms) is 100,000. Then we might expect 1000 of these terms to exceed this criterion, *even if the null hypothesis is true of all of them*.

This then suggests that we should set the criterion in relation to the size of the vocabulary, V . A threshold of $1/V$ would imply that we might expect (assuming the null hypothesis applies to all terms) a single noise term. Safety might suggest setting an even stricter threshold; however, since there will be many terms in the vocabulary that have a negative association with relevance, and many others that occur in one document only (which would give them no

chance of being selected on any such criterion), it is probably reasonable to relax the threshold somewhat. In the experiments, we have used a threshold $1/Ve^c$, for some constant c . Thus we may express the criterion as:

$$\left(\frac{n_t}{N}\right)^{r_t} \left(\frac{R}{r_t}\right) < \frac{1}{Ve^c}$$

or

$$r_t \log \frac{N}{n_t} - \log \left(\frac{R}{r_t}\right) - \log V > c \quad (5)$$

If the threshold c is set at 0, this is equivalent to taking a likelihood threshold of $1/V$, that is to accepting about one noise term under the simplest model as discussed above. A positive c is a stricter threshold; a value of 4.6 corresponds to having a less than 1% chance of accepting any noise terms. Experiments suggested that we could afford to relax the threshold; we used some negative c values.

5 Automatic adhoc and web tracks

The procedure for the official runs (Table 1) was as follows. Pools of potential expansion terms were generated by running pilot searches on terms extracted from the appropriate topic fields against the TREC disks 1–5 database, and outputting the top R documents. Terms extracted from these documents were weighted with the usual $w^{(1)}$ formula. In some cases an additional weighting was applied to topic terms. Terms were selected from the pools by means of the new selection procedure equation 5; two runs were done, using different term selection thresholds, for each topic source type (long, medium and short), and these were merged in pairs to produce the submitted runs. The thresholds varied between -4 and 4.6 ; there were other minor variations between runs, including for example the exact treatment of query terms.

Table 1: Automatic ad hoc and web track, official runs

Run	source	R	AveP					
			score	\geq med	P10	P30	RPrec	Rcl
ok8alx	TND	20	324	46	570	443	354	694
ok8amxc	TD	14	317	45	550	425	347	679
ok8wmxc	web, as ok8amxc		383	47	516	399	518	900
ok8asxc	T	10	279	32	488	380	309	637

Table 2 summarizes some diagnostic ad hoc runs, this time with single thresholds. It is not clear that the new term selection procedure offers any advantage over the old one; more investigation is needed. Removal of negative expressions was beneficial overall, increasing average precision in 17 topics and decreasing it in 10. However, trials on data from previous TRECs have shown that the effect averaged over 50 topics is not always beneficial.

6 Adaptive filtering

The system for adaptive filtering is very similar to the one used for TREC-7. It is described in [12] and more fully in a paper to appear next year [15]. A brief summary only will be given here. Please note that the equation for adapting the β value given in the TREC-7 proceedings contains a mistake; the correct version is given below and in the Journal of Documentation paper.

We assume the usual filtering situation, as constrained by the TREC filtering track rules. In particular, we assume an incoming stream of documents, a set of persistent user profiles (initially based on text topics), an accumulating collection of all the documents that have arrived so far, and a history for each profile, including relevance judgments for documents previously returned to the user. The entire process is switched on at time $t = 0$.

6.1 Calibration

The system used the usual BM25 match function, but the filtering task requires a threshold to be applied to the match values, to drive a retrieval decision. Thresholding in this system is tied to a calibration of the match function, to give values which can be regarded as actual probabilities of relevance. An initial calibration of the score is modified

Table 2: Ad hoc diagnostic runs

"Long topic" runs	Conditions	AveP	% gain	P30	% gain
ok8alpl1.neg	baseline (no expansion, no passages)	271	0.0	380	0.0
ok8alpl1	baseline + neg exprns removed + minor mods to parsing rules	277	2.2	386	1.6
ok8al2np.tns	+ exp 20 docs, eqn 5 term selection, threshold $c = 0.0$	310	14.4	427	12.4
ok8al2.tns	+ passages	327	20.6	437	15.0
ok8al2np.tns.rsv	Same # trms/query as ok8al2np.tns but trm sel by eqn 3	310	14.4	432	13.7
ok8al2np.tns.rsv.f51	as previous but fixed 51 terms per query	314	15.9	436	14.7
ok8al2np.tns.rsv.f30	as previous but fixed 30 terms per query	310	14.4	432	13.7
"Medium topic" runs	Conditions	AveP	% gain	P30	% gain
ok8ampl1	baseline (no expansion, no passages)	261	0.0	361	0.0
ok8aml1np.tns	+ exp 14 docs, eqn 5 term selection, threshold $c = -4.0$	304	16.5	415	15.0
ok8aml1.tns	+ passages	318	21.8	422	16.9
ok8aml1np.tns.rsv	Same # trms/query as ok8aml1np.tns but trm sel by eqn 3	298	14.2	411	13.9
ok8aml1np.tns.rsv.f55	as previous but fixed 55 terms per query	308	18.0	417	15.5
"Short topic" runs	Conditions	AveP	% gain	P30	% gain
ok8aspl1	baseline (no expansion, no passages)	239	0.0	343	0.0
ok8asl1np.tns	+ exp 10 docs, eqn 5 term selection, threshold $c = 3.0$	270	13.0	371	8.2
ok8asl1.tns	+ passages	279	16.7	377	9.9
ok8asl1np.tns.rsv	Same # trms/query as ok8asl1np.tns but trm sel by eqn 3	267	11.7	363	5.8
ok8asl1np.tns.rsv.f17	as previous but fixed 17 terms per query	288	20.5	383	11.7

on a per-topic basis, as feedback is obtained. The initial calibration is based on a logistic regression analysis of some training data. This gives values for β and γ in the equation:

$$\log O(R|D) = \beta + \gamma f(\text{score}, \text{ast1}, \text{maxscore}, ql) \quad (6)$$

$f(\text{score}, \dots)$ is basically a normalization function for the score, taking into account some variables which might be expected to affect it; two forms of $f()$ were used (see below). *maxscore* is the theoretical maximum score; *ql* is the query length in terms.

ast1 is the average score of the top 1% of retrieved documents; under TREC filtering rules (no access to any part of the document stream before $t = 0$), we do not initially have a direct method of estimating this, so another linear regression is performed on the same training data to estimate it for the first simulated week:

$$\text{ast1} = \alpha_1 + \alpha_2 \text{maxscore}$$

After the first week, *ast1* is estimated directly from the accumulated collection.

6.2 Score normalization

According to the arguments presented in the last TREC, it would be reasonable to assume that $f(\text{score}, \dots)$ should be linear in the score – that is, the score is assumed to be a linear function of the log-odds of relevance. However, this assumption depends on the basic independence assumptions of the probabilistic model, which may be doubted. A close look at the regression data suggested strongly that the relationship was non-linear, in particular that very high scores did not imply correspondingly high probabilities of relevance. We therefore tried a non-linear function as well as a linear one. Choice of functions was determined by the best-fitting logistic regression; we tried various linear and non-linear functions and chose the best of each after completing the regression. We also did the same exercise on three different training sets and took a guess at a compromise solution (this last step was relatively easy, as the solutions from the three training sets were generally very close).

The first step normalization was a linear one:

$$\text{normscore} = \frac{\text{score}}{\text{ast1} + 0.22\text{maxscore} - 1.3ql} \quad (7)$$

For the linear normalization we used exactly this, that is:

$$f(\text{score}, \dots) = \text{normscore}$$

For the non-linear normalization, we made a further transformation:

$$f(\text{score}, \dots) = \frac{\text{normscore}}{0.4 + \text{normscore}} \quad (8)$$

6.3 Adaptation

Given a document score and an estimated *ast1*, equation 6 can be used to estimate the log-odds of relevance of any specific document, to be compared to an absolute threshold determined by the desired utility measure. The result of applying the equation to the score for document D_i will be denoted c_i (for calibrated score). c_i is on a log-odds scale, but can be converted back to a probability p_i :

$$\begin{aligned} c_i &= \beta + \gamma f(\text{score}, \text{ast1}, \text{maxscore}, \text{ql}) \\ p_i &= \frac{\exp c_i}{1 + \exp c_i} \end{aligned} \quad (9)$$

for some estimated β , γ and *ast1*.

As we obtain feedback on the model, as well as re-estimating *ast1*, we adjust the calibration by correcting β (γ is left unchanged). β estimation is based on a Bayesian argument, in order to prevent it going wild with small amounts of data. The Bayesian prior is represented by m mythical documents whose estimated probabilities of relevance are assumed to be correct at 0.5. We suppose an iterative sequence of estimates $\beta^{(n)}$ and corresponding values $c_i^{(n)}$ and $p_i^{(n)}$ for each document. Then the gradient descent formula is:

$$\beta^{(n+1)} = \beta^{(n)} + \frac{r - \sum_{i=1}^j p_i^{(n)} + m \frac{1 - \exp(\beta^{(n)} - \beta^{(0)})}{2(1 + \exp(\beta^{(n)} - \beta^{(0)}))}}{\sum_{i=1}^j p_i^{(n)}(1 - p_i^{(n)}) + m \frac{\exp(\beta^{(n)} - \beta^{(0)})}{(1 + \exp(\beta^{(n)} - \beta^{(0)}))^2}} \quad (10)$$

$\beta^{(0)}$ is the initial estimate provided by the original regression equation 6.

(Please note that the first component of the denominator of this equation was incorrect in our TREC-7 report. The form we actually used was however correct.)

In the experiments, as last year, one iteration only was performed at each stage (simulated week), and only when new documents have been assessed; however, successive stages are cumulative, and at each stage all previously assessed documents are included in the estimation process. This procedure represents a very simple-minded approach to the normal iterative estimation suggested by the above argument.

Again as last year, we use a ‘ladder’ of thresholds, starting lower down in order to get some documents to the user for feedback purposes, even if this lowers performance in the early stages of the profile. The current ladder is given in table 3.

Table 3: The Ladder: selection thresholds

(Slightly modified from the TREC-7 version)

Initial points are explained in the text.

$P(R D)$	$\log O(R D)$	
0.5	0	
0.4	-0.4	Final (LF1 runs)
0.25	-1.1	Final (LF2 runs)
0.18	-1.5	First week
0.13	-1.9	
0.1	-2.2	High start
0.07	-2.6	
0.05	-2.9	
0.04	-3.2	Low start

In the circumstances of the TREC filtering task, an additional constraint applies: because the initial estimate (based on *maxscore* rather than on *ast1*) may be unreliable, and may in particular lead to the retrieval of many too many documents, the threshold is kept high for the first week. When a direct estimate of *ast1* becomes available, the

ladder is brought into effect, and the threshold is moved down to the appropriate place (possibly above the bottom if we found some documents in the first week).

Various other feedback methods may be brought into effect at various stages in the history of the profile. These include:

- Reweighting query terms
- Query expansion based on term selection value
- Query optimization (weights and/or selection of terms)
- Threshold optimization.

In general, any query adjustment has to be undertaken before any threshold setting, as it affects both *ast1* and the scores of the judged documents, all of which are used in threshold setting.

As last year, on this occasion we have tried only the threshold optimization. No term reweighting or query expansion methods were tried.

6.4 Experiments

Training

The training databases were: LA Times data with TREC topics 301–350; AP newswire data with topics 51–150; Wall Street Journal data with topics 51–150. All fields of the topics (Title, Description, Narrative) were used.

Each topic was searched on this collection, and the top ranked documents were retrieved, the number being specified as 1% of the collection. A series of logistic regression analyses were performed, to estimate β and γ in equation 6, with different functions for $f(\text{score}, \dots)$. The final choices for this function were those specified above (equations 7 and 8); the final choices for β and γ were -7.5 and 6.6 respectively for the linear score normalization, and -18.3 and 24.4 for the non-linear normalization.

The test topics themselves were also used in their entirety (title, description, narrative). They were initially searched on a database consisting of the three training databases merged, to establish initial terms and weights.

Adaptive procedure

Documents were processed in weekly batches. For the first week, the threshold was set at the point labelled 'First week' (because of the uncertainty of the *ast1* value). From the following week, a direct estimate of *ast1* is available, and two different initial points were tried (labelled 'High start' and 'Low start' in the table). Thereafter, the usual ladder rule applied: each profile was moved up one notch for each relevant document found. (As some profiles will have found relevant documents in the first week, these ones will never actually be at their theoretical starting point.) *ast1* is re-estimated from the accumulated collection for the first six weeks.

After each week in which some documents have been found for a profile (irrespective of relevance), the adaptive calibration of β is invoked. That is, for each previously seen document, a value of c_i is calculated according to the current profile and the current value of *ast1*, and one iteration only of the iterative formula 10 is then applied. The value of m in equation 10 was 5. The new value of β remains in force for this profile until the next invocation of the adaptive calibration.

Runs

Twelve candidate runs were completed, defined by the following parameters:

Treatment: 1 Base (Low start, no threshold optimization); 2 High start; 3 High start + Threshold optimization.

Utility function: 1 LF1; 2 LF2.

Score normalization: 1 Linear; 2 Non-linear.

Runs are numbered in the form ok8fxyz, where x is the treatment, y is the utility function, and z is the score normalization. Because of the limitation on the number of submissions, a random choice of six of these runs was submitted.

Results

Official results for the submitted runs, plus unofficial results for the others, are shown in Table 4. The table makes a topic-by-topic comparison with the other submissions to TREC, on utility scores. Some recall and precision results for all 12 runs (macro average), using the relevance judgments made for TREC-8, are shown in Table 5.

Table 4: Adaptive filtering: comparison with other TREC-8 official runs

Columns contain number of topics (out of 50)										
* unofficial results										
	All years					1994 only				
	Worst	< med	Median	> med	Best	Worst	< med	Median	> med	Best
	Utility function LF1 (out of 19 systems)									
ok8f111*	0	42	1	7	0					
ok8f112	0	43	3	4	0	0	25	5	20	6
ok8f211*	0	26	5	19	2					
ok8f212*	0	31	2	17	1					
ok8f311	0	23	6	21	2	0	13	14	23	9
ok8f312	0	27	5	18	1	0	15	10	25	9
	Utility function LF2 (out of 13 systems)									
ok8f121*	0	38	1	11	0					
ok8f122	23	39	4	7	0	16	26	7	17	4
ok8f221*	0	24	3	23	1					
ok8f222	1	28	3	19	2	4	24	10	16	3
ok8f321	0	25	4	21	2	2	17	14	19	5
ok8f322*	1	29	1	20	3					

Discussion of results

Overall the results are disappointing. This is partly the result of continuing to concentrate on the thresholding problem, at the expense of any query expansion/modification techniques. It is encouraging that our relative performance improves over the three simulated years of the task, even despite the lack of any query modification; the threshold adaptation methods are certainly valuable.

Starting higher up the ladder than the bottom (ok8f2xx) is clearly advantageous. As indicated at TREC-7, the ladder is very arbitrary, and it is entirely possible that we have overemphasized the argument about getting additional feedback material early. The non-linear normalization of score (ok8fxx2) had a very slight negative effect. Although it gives without doubt a better approximation to the probability of relevance, its only real effect is outside the range in which we are interested (in particular, the very top scoring documents), and it appears that the linear approximation is a quite good enough substitute for this purpose, and may even gain by being simpler. The threshold optimization seems to help a little.

7 VLC

Database processing

Before indexing, the source text was reduced by removing lines starting with "Server:", "Content-type:", "Last modified:", etc, document numbers were then identified, followed by the removal of all text inside '< ... >'. Dates and URLs were retained, but not indexed. This reduced the indexable text by almost 50% to a little over 50 GB. (Source text was unchanged from TREC-7.)

Examination of a few of the documents suggested that there was quite a lot of non-text material (compressed data etc). It was decided that it would not be practicable to remove (or avoid indexing) this material. This resulted in an index with a very large dictionary file of some 70 million terms most of which are nonsensical nonce-words, a typical sequence being "qetura", "qetutnz7", "qetuwwqgrslk79", "qetv", "qetv9pif0yk9",

Table 5: Recall and precision results

Average of ratios precision and recall results All years based on 49 topics (excluding one with no relevant documents) (none of these retrieved sets was empty)		
	Precision	Recall
ok8f111	0.174	0.338
ok8f112	0.160	0.343
ok8f121	0.168	0.362
ok8f122	0.151	0.375
ok8f211	0.239	0.257
ok8f212	0.227	0.274
ok8f221	0.220	0.295
ok8f222	0.203	0.326
ok8f311	0.248	0.252
ok8f312	0.238	0.265
ok8f321	0.236	0.271
ok8f322	0.226	0.289

The database was reindexed without positional information for TREC-8. The resulting index size was 14 GB (compared with 34 GB for the full index used in TREC-7). The total number of indexed tokens from the 18.6 million documents was about 5800 million (mean 312 per document), and the corresponding figure for types was 2600 million (140 per document).

Results

Table 6 summarizes the results. The official runs are ok8v1 and ok8v2. All runs used plain unexpanded ad hoc searches. The standard stop list used for ok8v1 contains 222 words. The extended stoplist used for all other runs contained also “i”, “inform..”, “does”, “me”, “find”; these were added with the intention of speeding searches, which they did. With normal “TREC-sized” databases and memory in the 256–512 MB range, BSS searches usually go more quickly if output sets are formed in (virtual) memory rather than explicitly on disk; but for the VLC much more physical memory would be required for this to hold. Finally, the BSS facility for heuristic limitation of output set size (which has been in use at least since TREC-3) has a marked effect when the required output is few documents from a large database.

Table 6: VLC results

Run	Conditions	Secs/query	Mod. AveP	P10	P30
ok8v1	no expansion, no passages	5.88	431	568	528
ok8v2	as v1 but slightly larger stoplist	4.30	445	560	538
ok8v23	as v2 but temp files instead of memory	3.82	445	560	538
ok8v22	as v23 but no output set size limitation	7.90	445	560	538

8 Discussion

The methods used in Okapi in general continue to give good results in several tracks. However, it is noticeable that of the three main modifications introduced this year (absolute expansion term selection, removal of negations, non-linear model for score calibration), only the one with a linguistic motivation seemed to help us.

It is clear that in order to achieve reasonable results in the filtering track, we need to move on to query expansion. However, the concentration on threshold setting has been useful, and the improvement in relative performance in the third year of the simulation probably indicates that it is doing something which other methods are failing to achieve.

References

- [1] Mitev, N.N., Venner, G.M. and Walker, S. *Designing an online public access catalogue: Okapi, a catalogue on a local area network*. British Library, 1985. (Library and Information Research Report 39.)
- [2] Walker, S. and Jones, R.M. *Improving subject retrieval in online catalogues: 1. Stemming, automatic spelling correction and cross-reference tables*. British Library, 1987. (British Library Research Paper 24.)
- [3] Walker, S. and De Vere, R. *Improving subject retrieval in online catalogues: 2. Relevance feedback and query expansion*. British Library, 1990. (British Library Research Paper 72.) ISBN 0-7123-3219-7
- [4] Robertson, S.E. *et al.* Okapi at TREC. In: [16], p21-30.
- [5] Robertson, S.E. and Sparck Jones K. Relevance weighting of search terms. *Journal of the American Society for Information Science* 27, May-June 1976, p129-146.
- [6] Robertson, S.E. *et al.* Okapi at TREC-2. In: [17], p21-34.
- [7] Robertson, S.E. *et al.* Okapi at TREC-3. In: [18], p109-126.
- [8] Robertson, S.E. *et al.* Okapi at TREC-4. In: [19], p73-96.
- [9] Beaulieu, M.M. *et al.* Okapi at TREC-5. In: [10], p143-165.
- [10] *The Fifth Text REtrieval Conference (TREC-5)*. Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 1997.
- [11] Walker S. *et al.* Okapi at TREC-6: Automatic ad hoc, VLC, routing, filtering and QSDR. In: [20], p125-136.
- [12] Robertson, S.E., Walker, S. and Beaulieu, M. Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive track. In: [21], p253-264.
- [13] Robertson, S.E. On term selection for query expansion. *Journal of Documentation* 46, Dec 1990, p359-364.
- [14] Porter, M.F. An algorithm for suffix stripping. *Program* 14 (3), Jul 1980, p130-137.
- [15] Robertson, S.E. and Walker, S. Threshold setting in adaptive filtering. *Journal of Documentation*, 56, 2000 (to appear).
- [16] *The First Text REtrieval Conference (TREC-1)*. Edited by D.K. Harman. Gaithersburg, MD: NIST, 1993.
- [17] *The Second Text REtrieval Conference (TREC-2)*. Edited by D.K. Harman. Gaithersburg, MD: NIST, 1994.
- [18] *Overview of the Third Text REtrieval Conference (TREC-3)*. Edited by D.K. Harman. Gaithersburg, MD: NIST, 1995.
- [19] *The Fourth Text REtrieval Conference (TREC-4)*. Edited by D.K. Harman. Gaithersburg, MD: NIST, 1996.
- [20] *The Sixth Text REtrieval Conference (TREC-6)*. Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 1998.
- [21] *The Seventh Text REtrieval Conference (TREC-7)*. Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 1999.

The Weaver System for Document Retrieval

Adam Berger and John Lafferty

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213

<aberger,lafferty>@cs.cmu.edu

Abstract

This paper introduces Weaver, a probabilistic document retrieval system under development at Carnegie Mellon University, and discusses its performance in the TREC-8 *ad hoc* evaluation. We begin by describing the architecture and philosophy of the Weaver system, which represents a departure from traditional approaches to retrieval. The central ingredient is a statistical model of how a user might distill or “translate” a given document into a query. The retrieval-as-translation approach is based on the noisy channel paradigm and statistical language modeling, and has much in common with other recently proposed models [12, 10]. After the initial high-level overview, the bulk of the paper contains a discussion of implementation details and the empirical performance of the Weaver retrieval system.

1 Introduction

This paper introduces the Weaver system for document retrieval, and discusses its performance on the TREC-8 *ad hoc* retrieval task. Weaver represents a significant departure from the traditional *tfidf*-based retrieval architecture, and its performance in TREC-8 suggests the promise of exploring new probabilistic approaches to retrieval.

The Weaver system is based on the use of statistical language modeling methods and the noisy channel paradigm from communication theory. At its core, however, is a model originally introduced in the context of statistical machine translation [5]. For this reason, we named the system after Warren Weaver, who nearly fifty years ago was the first to propose (albeit decades before computers were up to the task) that statistical techniques might be used to automate the process of translating text from one language into another.

This paper gives a brief overview of the guiding principles behind Weaver, and discusses several implementation details, including a description of how Weaver estimates the parameters of its statistical models from the TREC document collection. The following section contains an abbreviated discussion of the mathematical fundamentals behind Weaver; a more thorough treatment can be found in [1]. Section 3 provides details on the architecture of the Weaver system used in TREC-8. Section 4 contains information on the performance of Weaver in TREC-8, and some preliminary analysis of the results. We conclude in Section 5 by outlining some directions for future work.

2 The Probabilistic Framework

In formulating a query to a retrieval system, we imagine that a user begins with an information need. This information need is then represented as a fragment of an “ideal document”—a portion of the type of document that the user hopes to receive from the system. The user then

translates or distills this ideal document fragment into a succinct query, selecting key terms and replacing some terms with related terms.

Summarizing the model of query generation,

1. The user has an information need \mathfrak{S} .
2. From this need, he generates an ideal document fragment $\mathbf{d}_{\mathfrak{S}}$.
3. He selects a set of key terms from $\mathbf{d}_{\mathfrak{S}}$, and generates a query \mathbf{q} from this set.

One can view this imaginary process of query formulation as a corruption of the ideal document. In this setting, the task of a retrieval system is to find those documents most similar to $\mathbf{d}_{\mathfrak{S}}$. In other words, retrieval is the task of finding, among the documents comprising the collection, likely preimages of the user's query. Figure 1 depicts this model of retrieval in a block diagram.

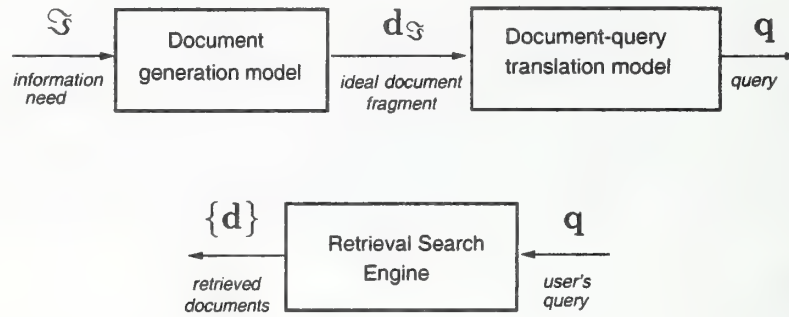


Figure 1. *Model of query generation and retrieval*

We have drawn Figure 1 in a way that suggests an information-theoretic perspective. One can view the information need \mathfrak{S} as a signal that gets corrupted as the user \mathcal{U} distills it into a query \mathbf{q} . That is, the query-formulation process represents a noisy channel, corrupting the information need just as a telephone cable corrupts the data transmitted by a modem. Given \mathbf{q} and a model of the channel—how an information need gets corrupted into a query—the retrieval system's task is to identify those documents \mathbf{d} that best satisfy the information need of the user.

More precisely, the retrieval system's task is to find the *a posteriori* most likely documents given the query; that is, those \mathbf{d} for which $p(\mathbf{d} | \mathbf{q}, \mathcal{U})$ is highest. By Bayes' law,

$$p(\mathbf{d} | \mathbf{q}, \mathcal{U}) = \frac{p(\mathbf{q} | \mathbf{d}, \mathcal{U}) p(\mathbf{d} | \mathcal{U})}{p(\mathbf{q} | \mathcal{U})}. \quad (1)$$

Since the denominator $p(\mathbf{q} | \mathcal{U})$ is fixed for a given query and user, we can ignore it for the purpose of ranking documents, and define the relevance $\rho_{\mathbf{q}}(\mathbf{d})$ of a document to a query as

$$\rho_{\mathbf{q}}(\mathbf{d}) = \underbrace{p(\mathbf{q} | \mathbf{d}, \mathcal{U})}_{\text{query-dependent}} \underbrace{p(\mathbf{d} | \mathcal{U})}_{\text{query-independent}}. \quad (2)$$

Equation (2) highlights the decomposition of relevance into two terms: first, a query-dependent term measuring the proximity of \mathbf{d} to \mathbf{q} , and second, a query-independent or "prior" term, measuring the quality of the document according to the user's general preferences and information needs.

The documents in the TREC-8 evaluation comprise a rather “well-behaved” collection, insofar as very few are completely implausible candidates for any possible query. Therefore, we expect there to be little harm in taking $p(\mathbf{q}|\mathcal{U})$ to be uniform. However, we imagine that in retrieval systems using real-world document collections, a non-uniform prior will be crucial for improved performance, and for adapting to the user’s needs and interests. At the very least, the document prior can be used to discount short documents, or perhaps documents in a foreign language.

We give a detailed formulation of one $p(\mathbf{q}|\mathbf{d})$ model below, but here we will briefly outline the strategy for constructing the model. We start with a corpus of (\mathbf{d}, \mathbf{q}) pairs, where each pair consists of a query and a document relevant to the query. Given this data, one can construct a translation model $p(\mathbf{q}|\mathbf{d})$, which assigns a probability to the event that \mathbf{q} is a distillation of (a translation of) \mathbf{d} .

Retrieval as translation

High-performance document retrieval systems must be sophisticated enough to handle synonymy and polysemy—to know, for instance, that **pontiff** and **pope** are related terms, and that **suit** can refer to clothing or a venue for legal grievance. The field of statistical translation concerns itself with how to mine large text databases to automatically discover such semantic relations. Brown *et al.* [4, 6] showed, for instance, how a system can “learn” to associate French terms with their English translations, given only a collection of bilingual French/English sentences. We shall demonstrate how, in a similar fashion, an IR system can, from a collection of documents, automatically learn which terms are related, and exploit these relations to better rank documents by relevance to a query.

By “translation model,” we mean a conditional probability distribution $p(\mathbf{f}|\mathbf{e})$ over sequences of source words $\mathbf{f} = \{f_1, \dots, f_m\}$, given a sequence of target words $\mathbf{e} = \{e_1, \dots, e_n\}$. In the context of French-to-English translation, the value $p(\mathbf{f}|\mathbf{e})$ is the probability that, when presented with the English word sentence \mathbf{e} , an expert translator will produce the French sequence \mathbf{f} .

Brown *et al.* [4] introduce the idea of an *alignment* \mathcal{A} between sequences of words, which captures how subsets of English words conspire to produce each French word. They also introduce the idea of a *null word*, an artificial word added to position zero of every English sentence, whose purpose is to generate those French words not strongly correlated with any other words in the English string.

Using \mathcal{A} , we can decompose $p(\mathbf{f}|\mathbf{e})$ as

$$p(\mathbf{f}|\mathbf{e}) = \sum_{\mathcal{A}} p(\mathbf{f}, \mathcal{A}|\mathbf{e}) = \sum_{\mathcal{A}} p(\mathbf{f}|\mathcal{A}, \mathbf{e}) p(\mathcal{A}|\mathbf{e}) \quad (3)$$

The IBM family of translation models is predicated on the simplifying assumption that exactly one English word is responsible for a given French word. We can therefore write

$$p(\mathbf{f}|\mathcal{A}, \mathbf{e}) = \prod_{i=1}^m t(f_i | e_{a_i}) \quad (4)$$

Here e_{a_i} is the English word aligned with the i th French word, and $t(f|e)$ is a parameter of the model—the probability that the English word e is paired with the French word f in the alignment.

We use the convention that boldface roman letters refer to collections of words such as documents or queries, while italic roman letters refer to individual terms. Thus $p(\mathbf{q}|\mathbf{d})$ refers to the probability of generating a *single* query word from an entire document \mathbf{d} .

If \mathbf{f} contains m words and \mathbf{e} contains $n + 1$ words (including the null word), there are $(n + 1)^m$ alignments between \mathbf{e} and \mathbf{f} . The most basic member of this family of models, Model 1, simplifies matters dramatically by assuming that the translation probability $p(\mathbf{f} | \mathbf{e})$ does not depend on the order in which the words appear in the sentences. Thus we can write

$$p(\mathbf{f} | \mathbf{e}) = \frac{p(m | \mathbf{e})}{(n + 1)^m} \sum_A \prod_{i=1}^m t(f_i | e_{a_i}). \quad (5)$$

Given a collection of bilingual sentences $\mathcal{C} = \{(\mathbf{f}_1, \mathbf{e}_1), (\mathbf{f}_2, \mathbf{e}_2), (\mathbf{f}_3, \mathbf{e}_3) \dots\}$, the likelihood method suggests that one should adjust the parameters of (5) in such a way that the model assigns as high a probability as possible to \mathcal{C} . This maximization must be performed, of course, subject to the constraints $\sum_f t(f | e) = 1$ for all e . Using Lagrange multipliers,

$$t(f | e) = \lambda^{-1} \sum_A p(\mathbf{f}, \mathcal{A} | \mathbf{e}) \sum_{j=1}^m \delta(f, f_j) \delta(e, e_{a_j}), \quad (6)$$

where δ is the Kronecker delta function.

The parameter $t(f | e)$ appears explicitly in the lefthand side of (6), and implicitly in the right. By repeatedly solving this equation for all pairs f, e (in other words, applying the EM algorithm), one eventually reaches a stationary point of the likelihood.

Equation (6) contains a sum over alignments, which is exponential and suggests that the computing the parameters in this way is infeasible. In fact, this is not the case, since

$$\sum_A \prod_{i=1}^m t(f_i | e_{a_i}) = \prod_{i=1}^m \sum_{j=1}^n t(f_i | e_j) \quad (7)$$

This rearranging means that computing $\sum_A p(\mathbf{f}, \mathcal{A} | \mathbf{e})$ requires only $\Theta(mn)$ work, rather than $\Theta(n^m)$.

Brown *et al.* propose a series of increasingly complex and powerful statistical models of translation, the parameters of which are estimated by a bootstrapping procedure. We have described here only that portion of the IBM translation approach that is directly relevant to the retrieval method described below. For further details on statistical machine translation, we refer the reader to two articles [4, 6].

A Model of Document-Query Translation

Suppose that an information analyst is given a news article and asked to quickly generate a list of a few words to serve as a rough summary of the article's topic. As the analyst rapidly skims the story, he encounters a collection of words and phrases. Many of these are rejected as irrelevant, but his eyes rest on certain key terms as he decides how to render them in the summary. For example, when presented with an article about Pope John Paul II's visit to Cuba in 1998, the analyst decides that the words **pontiff** and **vatican** can simply be represented by the word **pope**, and that **cuba**, **castro** and **island** can be collectively referred to as **cuba**.

We consider the simplest of the IBM translation models for the document-to-query mapping. This model produces a query according to the following generative procedure. First we choose a length m for the query, according to the distribution $\psi(m | \mathbf{d})$. Then, for each position $j \in [1 \dots m]$ in the query, we choose a position i in the document from which to generate q_j , and generate the query word by "translating" d_i according to the translation model $t(\cdot | d_i)$. We include in position zero of the document an artificial "null word," written **<null>**. The

purpose of the null word is to generate spurious or content-free terms in the query (consider, for example, a query $\mathbf{q} = \text{Find all of the documents...}$).

Let's now denote the length of the document by $|\mathbf{d}| = n$. The probability $p(\mathbf{q} | \mathbf{d})$ is then the sum over all possible alignments, given by

$$p(\mathbf{q} | \mathbf{d}) = \frac{\psi(m | \mathbf{d})}{(n+1)^m} \sum_{a_1=0}^n \cdots \sum_{a_m=0}^n \prod_{j=1}^m t(q_j | d_{a_j}). \quad (8)$$

Just as the most primitive version of IBM's translation model takes no account of the subtler aspects of language translation, including the way word order tends to differ across languages, so our basic IR translation approach is but an impressionistic model of the relation between queries and documents relevant to them. Since IBM called their most basic scheme Model 1, we shall do the same for this rudimentary retrieval model.

A little algebraic manipulation shows that the probability of generating query \mathbf{q} according to Model 1 can be rewritten as

$$p(\mathbf{q} | \mathbf{d}) = \psi(m | \mathbf{d}) \prod_{j=1}^m \left(\frac{n}{n+1} p(q_j | \mathbf{d}) + \frac{1}{n+1} t(w | \langle \text{null} \rangle) \right)$$

where

$$p(q_j | \mathbf{d}) = \sum_w t(q_j | w) l(w | \mathbf{d}),$$

with the *document language model* $l(w | \mathbf{d})$ given by relative counts. Thus, we see that the query terms are generated using a *mixture model*—the document language model provides the mixing weights for the *translation model*, which has parameters $t(q | w)$. An alternative view (and terminology) for this model is to describe it as a Hidden Markov Model, where the states correspond to the words in the vocabulary, and the transition probabilities between states are proportional to the word frequencies.

3 Architecture of Weaver

As described in the previous section, Weaver assigns relevance rankings to documents according to a probability $p(q | d)$ that a user would distill the document into the query. To rank documents, Weaver doesn't employ a reverse index, but instead visits each document d in the collection and computes (8) for each. This is a tremendously expensive operation, but one can reduce the work somewhat by first performing a *fast match*: eliminate all documents which share no words in common with the query.

Phrases

Although Weaver takes a non-traditional approach to retrieval, it still relies on a standard independence or "bag of words" assumption, ignoring word order within documents and queries. As a modest step in the direction of context-awareness, Weaver does recognize a select set of two-word phrases.

We identified phrases using the statistical measure of mutual information. Within the corpus \mathcal{C} of documents appearing on TREC disks 4 and 5 (excluding the Congressional Record documents), we ranked all pairs of words x, y according to their mutual information, and identified the highest-scoring pairs as phrases. In this context, the mutual information between two words is the reduction in uncertainty about the presence of y that results from knowing whether x was the preceding word.

FINANCI TIME	SAN DIEGO
UNIT STATE	WHITE HOUS
WALL STREET	FISCAL YEAR
PRIME MINIST	GEORG BUSH
SOVIET UNION	PRIVAT SECTOR
HONG KONG	NUCLEAR POWER
SAN JOSE	CHIEF EXECUTIVE
STOCK MARKET	PENSION FUND
SAN FRANCISCO	BILL CLINTON

Table 1: A subset of the 11,644 phrases automatically discovered from the newswire documents in TREC disks 4 and 5 using the mutual information criterion.

Define

$$p(y) = \frac{\text{count}(y)}{\sum_w \text{count}(w)}$$

$$p(x, y) = \frac{\text{count}(x, y)}{\sum_{v, w} \text{count}(v, w)}$$

as the frequency of the word y and the frequency of the bigram x, y , respectively. Furthermore, define

$$H(y) = -p(y) \log p(y) - (1 - p(y)) \log(1 - p(y))$$

$$H(y | x) = -p(x, y) \log \frac{p(x, y)}{p(x)} - (1 - p(x, y)) \log \left(1 - \frac{p(x, y)}{p(x)}\right)$$

as the *entropy* of the word y and the bigram x, y respectively. Putting these definitions together, the mutual information score of the bigram x, y is

$$I(x; y) \stackrel{\text{def}}{=} H(y) - H(y | x)$$

Intuitively, $I(x; y)$ measures the reduction in uncertainty about whether y will be the next word in a sequence of text, given that x was the previous word. In practice, bigrams like **Hong Kong** tend to exhibit high mutual information. Table 3 lists a selected subset of the 11,644 phrases automatically extracted from the *News* portion of TREC disks 4 and 5.

Vocabulary issues

We elected to use the Porter stemmer to canonicalize English surface forms. This stemmer's deficiencies are well known—it aggressively conflates words, a characteristic which can sometimes be a liability: **policy** and **police**, for instance, are mapped to the same stem. On this first large-scale trial of Weaver, however, we decided to err on the side of a smaller active vocabulary.

We employed the 571-member SMART stopword list to eliminate common words from documents and topics. After applying stemming, pruning stopwords, and adding statistical phrases, we partitioned the collection of active documents into two portions: Federal Register documents and news documents, as summarized in Table 3. We selected the most common 100,000 words from each corpus and built separate models on each corpus.

When searching for relevant documents, we scored Federal Register documents according to the models trained on this portion of the data and news documents according to their own model. Combining rankings across the two partitions was a simple matter: each model produces a probability estimate $p(\mathbf{q} | \mathbf{d})$, and these estimates are comparable across models.

model	corpus	size (in documents)
<i>FR</i>	<i>Federal Register (1994)</i>	55,630
<i>News</i>	<i>Financial Times (1991-1994)</i> <i>Los Angeles Times</i> <i>Foreign Broadcast Information Service</i>	472,525

Table 2: Since the content (and presumably occurrence statistics) of the Federal Register data appeared markedly different from the rest of the TREC-8 collection, we elected to separate out this data and process it independently of the rest.

Synthetic data

The translation model is parametrized in terms of $t(q | w)$, the probability that a word w in a document will “generate” the word q in a query for which that document is relevant. Before using these models, one needs to assign a value to each of these parameters. We compute maximum-likelihood values for the model parameters from a collection of queries and documents relevant to those queries. Given such a model and a new query \mathbf{q}' , assigning relevance judgments is a matter of computing $p(\mathbf{q}' | \mathbf{d})$ for each $\mathbf{d} \in \mathcal{C}$.

One could imagine using relevance judgments from previous TREC evaluations as the training data from which to learn model parameters. However, the number of parameters in the models we use is sufficiently large (the square of the number of recognized words) that just a few hundred topics won’t suffice to estimate the parameters accurately. In fact, we know of no publicly-available collection of relevance judgments of suitable heft. Therefore, we synthesize training data as follows: from a TREC document, select words randomly to create a query, and take the document to be relevant to the query. For details on generating synthetic data by sampling, we refer the reader to [1].

For the TREC-8 experiments, we generated, for each of the two partitions of the data, one million synthetic queries of 15 words each.

System configuration

We ran the TREC evaluation on one of six UltraSPARC II 248 Mhz processors belonging to a Sun UltraEnterprise 3000 machine containing 1.5GB of physical memory, running the SunOS 5.5.1. operating system. Other than the fast match described above, we performed no optimization for speed or memory usage. Parameter estimation and document ranking required several days to complete.

Smoothing

For statistical models of this form, *smoothing* or interpolating the parameters away from their maximum likelihood estimates is crucial. We used a simple linear mixture of the background unigram model and the EM-trained translation model:

$$p_{\alpha}(q | \mathbf{d}) = \alpha p(q | \mathcal{D}) + (1 - \alpha) p(q | \mathbf{d})$$

$$= \alpha p(q | \mathcal{D}) + (1 - \alpha) \sum_{w \in \mathbf{d}} l(w | \mathbf{d}) t(q | w).$$

The weight was empirically set to $\alpha = 0.05$ by optimizing performance on a different dataset: a portion of the 1998 TREC *Spoken Document Retrieval* (SDR) data. Figure 3 shows the behavior of the system on the TREC-7 evaluation as a function of α .

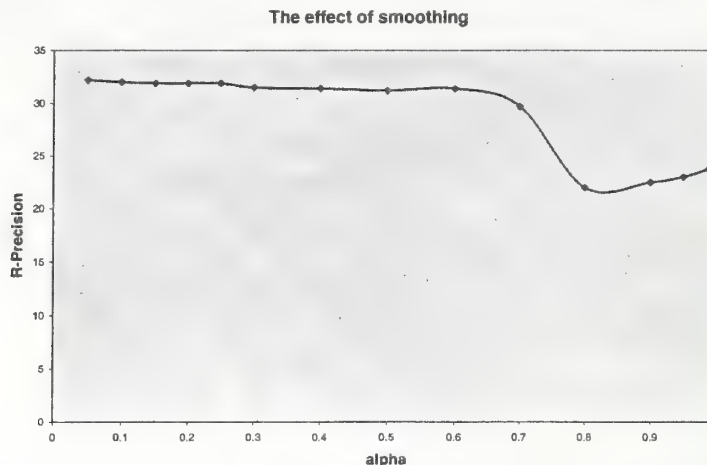


Figure 1: Retrieval performance of Weaver on the TREC-7 ad hoc retrieval task, as a function of α , the weight of the corpus-wide (“back-off”) unigram language model in $p(\mathbf{q} | \mathbf{d})$. Although we set $\alpha = 0.05$ for the TREC-8 evaluation, it appears that the system is quite insensitive to the exact degree of smoothing, at least within a reasonable range.

4 TREC-8 Performance

Table 4 contains the precision/recall results for the Weaver system within the TREC-8 ad hoc automatic evaluation, as reported by NIST. Figures 2 and 3 display the performance of Weaver relative to all other systems participating in the TREC-8 ad hoc automatic evaluation.

Precision:		Precision at:	
0.00	0.6426	5 docs:	0.4480
0.10	0.4673	10 docs:	0.4120
0.20	0.3671	15 docs:	0.3600
0.30	0.3179	20 docs:	0.3370
0.40	0.2804	30 docs:	0.3100
0.50	0.2363	100 docs:	0.2134
0.60	0.1887	200 docs:	0.1554
0.70	0.1574	500 docs:	0.0940
0.80	0.1157	1000 docs:	0.0588
0.90	0.0753	R-Precision:	0.2696
1.00	0.0355		
Average :	0.2447		

Table 3: Performance of the Weaver system in the TREC-8 automatic ad hoc evaluation. The topics consisted of titles and descriptions.

We report our results on the topics consisting of titles and descriptions only, but note in passing that the system performed slightly worse (roughly two percentage points in overall

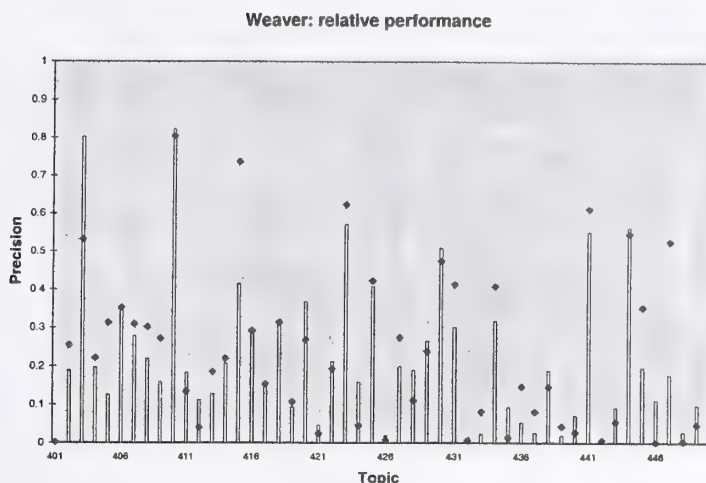


Figure 2: Performance of the Weaver system (represented by points) relative to the median performance of all automatic ad hoc TREC-8 systems (represented by vertical lines).

precision) when provided with the narrative portion as well. This is a somewhat unsuspected result—we observed the narratives to aid performance in internal experiments on earlier TREC datasets—which we plan to explore further.

5 Conclusions

TREC-8 marks the first large-scale evaluation of the retrieval-as-translation paradigm. Along with the language modelling approach put forward by the University of Massachusetts [12] and the Hidden Markov Model system deployed by BBN in TREC-7 [10], Weaver represents a departure from the traditional *tfidf*-based retrieval architecture. Its major asset is a strong theoretical grounding in probability; its major weakness is the computational burden it incurs.

Our immediate future plans will focus on ways to reduce this burden without compromising accuracy. Equipped with a faster retrieval engine, we hope to be able to conduct more experiments to understand how best to extend and improve the Weaver system. We are also investigating ways to incorporate pseudo-feedback into the probabilistic framework.

Acknowledgements

This research was supported in part by NSF KDI grant IIS-9873009, an IBM University Partnership Award, an IBM Cooperative Fellowship, and a grant from Justsystem Corporation. The first author gratefully acknowledges the support of Claritech Corporation throughout the TREC-8 evaluation.

References

- [1] A. Berger and J. Lafferty (1999). “Information retrieval as statistical translation,” *Proceedings of the ACM SIGIR*, pp. 222-229.

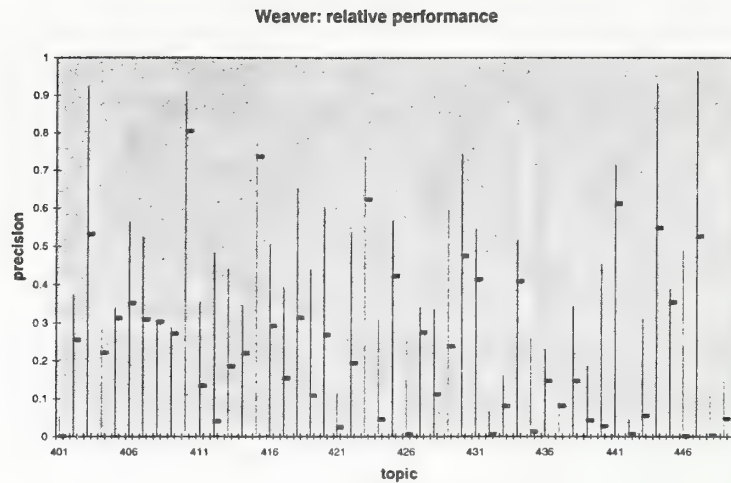


Figure 3: Performance of the Weaver system (represented by hash marks) relative to the best and worst performances among all automatic ad hoc TREC-8 systems.

- [2] A. Bookstein and D. Swanson (1974). "Probabilistic models for automatic indexing," *Journal of the American Society for Information Science*, **25**, pp. 312–318.
- [3] A. Broder and M. Henzinger (1998). "Information retrieval on the web: Tools and algorithmic issues," Invited tutorial at Foundations of Computer Science (FOCS).
- [4] P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin (1990). "A statistical approach to machine translation," *Computational Linguistics*, **16**(2), pp. 79–85.
- [5] A. Berger, P. Brown, S. Della Pietra, V. Della Pietra, J. Gillett, J. Lafferty, H. Printz and L. Ures (1994) "The Candide system for machine translation" *Proceedings of the ARPA Human Language Technology Workshop*, Plainsborough, New Jersey.
- [6] P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer (1993). "The mathematics of statistical machine translation: Parameter estimation," *Computational Linguistics*, **19**(2), pp. 263–311.
- [7] W. B. Croft and D. J. Harper (1979). "Using probabilistic models of document retrieval without relevance information," *Journal of Documentation*, **35**, pp. 285–295.
- [8] A. Dempster, N. Laird, and D. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, **39**(B), pp. 1–38.
- [9] W. Gale and K. Church (1991). "Identifying word correspondences in parallel texts," in *Fourth DARPA Workshop on Speech and Natural Language*, Morgan Kaufmann Publishers, pp. 152–157.
- [10] D. Miller, T. Leek and R. Schwartz (1999). "BBN at TREC-7: Using Hidden Markov Models for Information Retrieval," In *Proceedings of the Text REtrieval Conference (TREC-7)*, Gaithersburg, Maryland.
- [11] J. Ponte (1998). *A language modeling approach to information retrieval*. Ph.D. thesis, University of Massachusetts at Amherst.
- [12] J. Ponte and W. B. Croft (1998). "A language modeling approach to information retrieval," *Proceedings of the ACM SIGIR*, pp. 275–281.

- [13] S.E. Robertson and K. Sparck Jones (1976). "Relevance weighting of search terms," *Journal of the American Society for Information Science*, **27**, pp. 129–146.
- [14] S. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau (1992). "Okapi at TREC," In *Proceedings of the Text REtrieval Conference (TREC-1)*, Gaithersburg, Maryland.
- [15] G. Salton and C. Buckley (1988). "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, **24**, pp. 513–523.
- [16] H. Turtle and W.B. Croft (1991). "Efficient probabilistic inference for text retrieval," *Proceedings of RIAO 3*.
- [17] W. Weaver (1955). "Translation (1949)," In *Machine Translation of Languages*, MIT Press.

LASSO: A Tool for Surfing the Answer Net

Dan Moldovan, Sanda Harabagiu,
Marius Paşca, Rada Mihalcea, Richard Goodrum, Roxana Gîrju and Vasile Rus
Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275-0122
{moldovan,sanda,mars,rada,goodrum,roxana,rus}@seas.smu.edu

Abstract

This paper presents the architecture, operation and results obtained with the LASSO system developed in the Natural Language Processing Laboratory at SMU. The system relies on a combination of syntactic and semantic techniques, and lightweight abductive inference to find answers. The search for the answer is based on a novel form of indexing called paragraph indexing. A score of 55.5% for short answers and 64.5% for long answers was achieved.

Background

Finding the answer to a question by returning a small fragment of a text, where the answer actually lies, is profoundly different from the task of information retrieval (IR) or information extraction (IE). Current IR systems allow us to locate full documents that might contain the pertinent information, leaving it to the user to extract the answer from a ranked list of texts. In contrast, IE systems extract the information of interest, provided it has been presented in a predefined, target representation, known as *template*. The immediate solution of combining IR and IE techniques for question/answering (Q/A) is impractical, since IE systems are known to be highly dependent on domain knowledge, and furthermore, the generation of templates is not performed automatically.

Our methodology of finding answers in large collections of documents relies on natural language processing (NLP) techniques in novel ways. First, we perform a processing of the question by combining syntactic information, resulting from a shallow parse, with semantic information that characterizes the question (e.g. *question type*, *question focus*). Secondly, the search for the answer is based on a novel form of indexing, called *paragraph indexing* and new related retrieval methods. Finally, in order to extract the answers and to evaluate their correctness, we use a battery of abductive techniques, some based on empirical methods, some on lexico-semantic information. The principles that have

guided our paragraph indexing and the abductive inference of the answers are reported in (Harabagiu and Maiorano 1999).

When designing LASSO, the Q/A system developed by the NLP group at SMU, our goal was not to employ NLP techniques just for enhancing the IR results. Instead, we developed a Q/A model that retains the elegance of IR systems, by using shallow processing, and adds the exactness of IE systems, by providing with methods of finding and extracting answers without deep NLP. Furthermore, to comply with the open-domain constraints of the TREC Q/A task, we relied only on lexico-semantic resources that are of general nature. This design allows the escalation to Q/A systems capable of handling questions that impose high-level reasoning techniques (e.g. questions used in the evaluations of the *High Performance Knowledge Bases (HPKB)* program (Cohen et al.1998).

Overview of the LASSO Q/A System

The architecture of LASSO comprises three modules: *Question Processing* module, *Paragraph Indexing* module and *Answer Processing* module. Given a question, of open-ended nature, expressed in natural language, we first process the question by creating a representation of the information requested. Thus we automatically find (a) what *type of question* it is, from the taxonomy of questions at hand, (b) what *type of answer* is expected, and most importantly, (c) what is the *question focus* defined as the main information required by the interrogation. Furthermore, the *Question Processing* also identifies the keywords from the question, which are passed to the *Paragraph Indexing* module, as illustrated by Figure 1.

In LASSO, documents are indexed by a modified Zprise IR system available from NIST. Our search engine incorporates a set of Boolean operators (e.g. AND, OR, NOT, NEAR). We post-process the results of the IR search engine by filtering out the returns that do not contain all keywords in the same paragraph. This op-

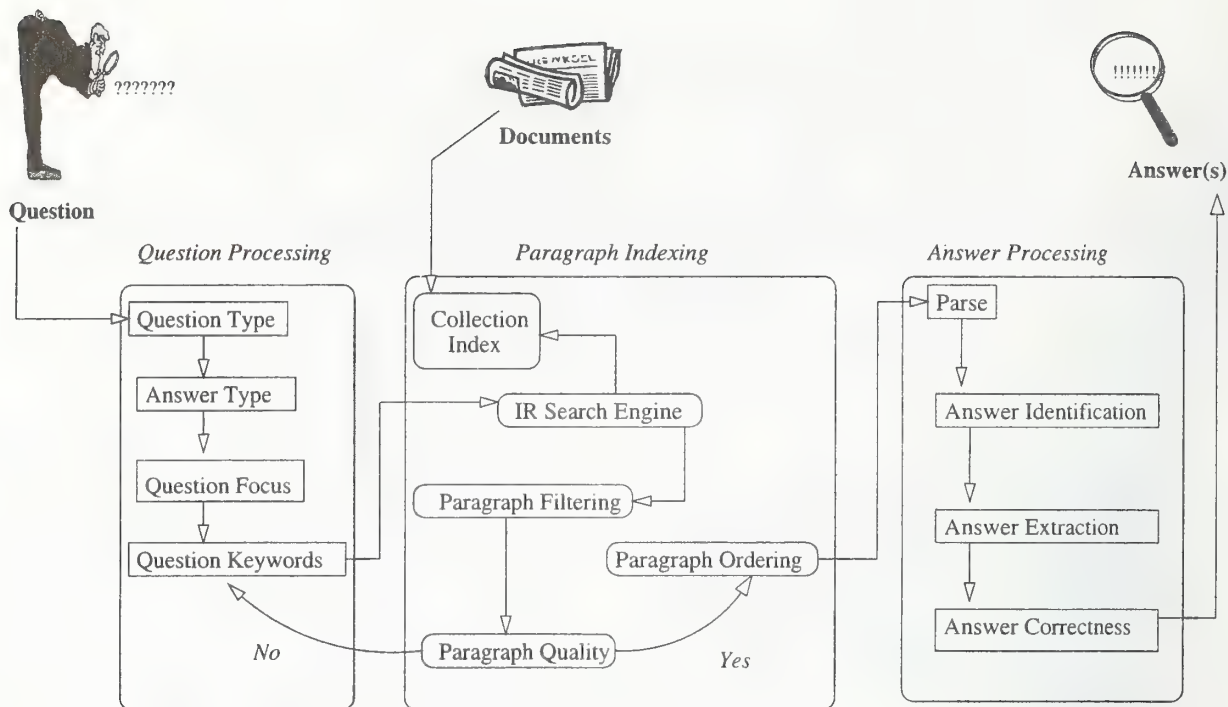


Figure 1: Architecture of the LASSO Q/A System

eration allows for on-the-fly generation of a paragraph index. The second important feature of the *Paragraph Indexing* module comes from the evaluation of the quality of the paragraphs. When the quality is satisfactory, we order the paragraphs according with a plausibility degree of containing the answer. Otherwise, we add/drop keywords and resume the paragraph retrieval. This loop generates a feed-back retrieval context that enables only a reasonable number of paragraphs to be passed to the *Answer Processing* module.

The advantage of processing paragraphs instead of full documents determines a faster syntactic parsing. Our parses also involve Named Entity recognitions and use of lexico-semantic resources that are valuable in the extraction of the answer. The extraction and evaluation of the answer correctness is based on empirical abduction.

Question Processing

The role of the question processing module is to: (1) determine the type of question, (2) determine the type of answer expected, (3) build a focus for the answer, and (4) transform the question into queries for the search engine.

In order to find the right answer to a question from a large collection of texts, first we have to know what we should look for. The answer type can usually be

determined from the question. For a better detection, of the answer, the questions are first classified by their type: *what*, *why*, *who*, *how*, *where* questions, etc. A further classification follows to better identify the question type. Table 1 shows the classification for the 200 TREC-8 questions. ”

We further realized that the question type was not sufficient for finding answers. For the questions like *Who was the first American in space?*, the answer type is obvious: PERSON. However, this does not apply for example to the questions of type *what*, as *what* is ambiguous and it says nothing about the information asked by the question. The same applies to many other question types. The problem was solved by defining a concept named *focus*.

A *focus* is a word or a sequence of words which define the question and disambiguate it in the sense that it indicates what the question is looking for, or what the question is all about. For example, for the question *What is the largest city in Germany?*, the focus is *largest city*. Knowing the focus and the question type it becomes easier to determine the type of the answer sought, namely: the name of the largest city in Germany.

The focus is also important in determining the list of keywords for query formation. We noticed that some

Q-class	Q-subclass	Nr. Q	Nr. Q answered	Answer type	Example of question	Focus
what		64	54			
	basic what	40	34	MONEY/NUMBER/DEFINITION/TITLE/NNP/UNDEFINED	What was the monetary value of the Nobel Peace Prize in 1989?	monetary value
	what-who	7	7	PERSON/ORGANIZATION	What costume designer decided that Michael Jackson should only wear one glove?	costume designer
	what-when	3	2	DATE	In what year did Ireland elect its first woman president?	year
	what-where	14	12	LOCATION	What is the capital of Uruguay?	capital
who		47	37	PERSON/ORGANIZATION	Who is the author of the book "The Iron Lady: A Biography of Margaret Thatcher"?	author
how		31	21			
	basic how	1	0	MANER	How did Socrates die?	Socrates
	how-many	18	13	NUMBER	How many people died when the Estonia sank in 1994?	people
	how-long	2	2	TIME/DISTANCE	How long does it take to travel from Tokyo to Niigata?	-
	how-much	3	2	MONEY/PRICE	How much did Mercury spend on advertising in 1993?	Mercury
	how-much-<modifier>	1	0	UNDEFINED	How much stronger is the new vitreous carbon material invented by the Tokyo Institute of Technology compared with the material made from cellulose?	new vitreous carbon material
	how-far	1	1	DISTANCE	How far is Yaroslavl from Moscow?	Yaroslavl
	how-tall	3	3	NUMBER	How tall is Mt. Everest?	Mt. Everest
	how-rich	1	0	UNDEFINED	How rich is Bill Gates?	Bill Gates
	how-large	1	0	NUMBER	How large is the Arctic refuge to preserve unique wildlife and wilderness value on Alaska's north coast?	Arctic refuge
where		22	16	LOCATION	Where is Taj Mahal?	Taj Mahal
when		19	13	DATE	When did the Jurassic Period end?	Jurassic Period
which		10	8			
	which-who	1	1	PERSON	Which former Klu Klux Klan member won an elected office in the U.S.?	former Klu Klux Klan member
	which-where	4	3	LOCATION	Which city has the oldest relationship as sister-city with Los Angeles?	city
	which-when	1	1	DATE	In which year was New Zealand excluded from the ANZUS alliance?	year
	which-what	4	3	NNP/ORGANIZATION	Which Japanese car maker had its biggest percentage of sale in the domestic market?	Japanese car maker
name		4	4			
	name-who	2	2	PERSON/ORGANIZATION	Name the designer of the show that spawned millions of plastic imitations, known as "jellies"?	designer
	name-where	1	1	LOCATION	Name a country that is developing a magnetic levitation railway system?	country
	name-what	1	1	TITLE/NNP	Name a film that has won the Golden Bear in the Berlin Film Festival?	film
why		2	0	REASON	Why did David Koresh ask for a word processor?	David Koresh
whom		1	0	PERSON/ORGANIZATION	Whom did the Chicago Bulls beat in the 1993 championship?	Chicago Bulls
Total		200	153 77%			

Table 1: Types of questions and statistics. In this table we considered that a question was answered correctly if its answer was among top five ranked long answers.

words in the questions never occur in the answer, and that is because their role is just to disambiguate the question. For example, in the question *In 1990, what day of the week did Christmas fall on?*, the focus is *day of the week*, a concept that is unlikely to occur in the answer. In such situations, the focus should not be included in the list of keywords considered for detecting the answer.

The process of extracting keywords is based on a set of ordered heuristics. Each heuristic returns a set of keywords, that is added in the same order to the question keywords. We have implemented eight different heuristics. Initially, only the keywords returned by the first six heuristics are considered. If further keywords are needed in the retrieval loop, keywords provided by the other two heuristics are added. When keywords define an exceedingly specific query, they are dropped in the reversed order in which they have been entered. The heuristics are:

- *Keyword-Heuristic 1:* Whenever quoted expressions are recognized in a question, all non-stop words of the quotation became keywords.
- *Keyword-Heuristic 2:* All named entities, recognized as proper nouns, are selected as keywords.
- *Keyword-Heuristic 3:* All complex nominals and their adjectival modifiers are selected as keywords.
- *Keyword-Heuristic 4:* All other complex nominals are selected as keywords.
- *Keyword-Heuristic 5:* All nouns and their adjectival modifiers are selected as keywords.
- *Keyword-Heuristic 6:* All the other nouns recognized in the question are selected as keywords.
- *Keyword-Heuristic 7:* All verbs from the question are selected as keywords.
- *Keyword-Heuristic 8:* The question focus is added to the keywords.

Table 2 lists two questions from the TREC-8 competition together with their associated keywords. The Table also illustrates the trace of keywords until the paragraphs containing the answer were found. For question 26, the paragraphs containing the answers could not be found before dropping many of the initial keywords. In contrast, the answer for question 13 was found when the verb *rent* was added to the Boolean query.

Paragraph Indexing

Search engine

The Information Retrieval Engine for LASSO is related to the Zprise IR search engine available from NIST. There were several features of the Zprise IR engine which were not conducive to working within the design of LASSO. Because of this, a new IR engine was generated to support LASSO without the encumbrance

Q-26	<i>What is the name of the "female" counterpart to El Nino, which results in cooling temperatures and very dry weather ?</i>
Keys	female El Nino dry weather cooling temperatures female El Nino dry weather cooling female El Nino dry weather female El Nino dry female El Nino female El
Q-13	<i>How much could you rent a Volkswagen bug for in 1966 ?</i>
Keys	Volkswagen bug Volkswagen bug rent

Table 2: Examples of TREC-8 Question Keywords

of these features. The index creation was, however, kept in its entirety.

The Zprise IR engine was built using a cosine vector space model. This model does not allow for extraction of those documents which include all of the keywords, but extracts documents according to the similarity measure between the document and the query as computed by the cosine of the angle between the vectors represented by the document and the query. This permits documents to be retrieved when only one of the keywords is present. Additionally, the keywords present in one retrieved document may not be present in another retrieved document.

LASSO's requirements are much more rigid. LASSO requires that documents be retrieved only when all of the keywords are present in the document. Thus, it became necessary to implement a more precise determinant for extraction. For the early work, it was determined that a Boolean discriminate would suffice provided that the operators AND and OR were implemented. It was also necessary to provide the ability to organize queries through the use of parentheses.

We opted for the Boolean indexing as opposed to vector indexing because Boolean indexing increases the *recall* at the expense of *precision*. That works well for us since we control the retrieval precision with the PARAGRAPH operator which provides document filtering. In addition, the Boolean indexing requires less processing time than vector indexing, and this becomes important when the collection size increases.

To facilitate the identification of the document sources, the engine was required to put the document id in front of each line in the document.

The index creation includes the following steps: normalize the SGML tags, eliminate extraneous characters, identify the words within each document, stem the terms (words) using the Porter stemming algorithm, calculate the local (document) and global (col-

lection) weights, build a comprehensive dictionary of the collection, and create the inverted index file.

The index generation process for LASSO is the same process as used by Zprise, however, several minor changes were necessary for the inclusion of the data presented.

It was observed that while the Zprise index process should work for multiple databases, it did not. Since there are four distinct sources of data present in the collection, four unique indices were created (one for each source). The fact that LASSO uses a Boolean discriminate versus a cosine vector space similarity measure makes this permissible. Furthermore, it became necessary to expand the number of SGML tags that are included in the index creation process. This was necessary since each source chose to use a different, but overlapping, set of SGML tags.

Paragraph filtering

The number of documents that contain the keywords returned by the Search Engine may be large since only weak Boolean operators were used. A new, more restrictive operator was introduced: PARAGRAPH *n*. This operator searches like an AND operator for the words in the query with the constraint that the words belong only to some *n* consecutive paragraphs, where *n* is a controllable positive integer.

The parameter *n* selects the number of paragraphs, thus controlling the size of the text retrieved from a document considered relevant. The rationale is that most likely the information requested is found in a few paragraphs rather than being dispersed over an entire document.

In order to apply this new operator, the documents retrieved by the search engine have to be segmented into sentences and paragraphs. Separating a text into sentences proves to be an easy task, one could just make use of the punctuation to solve this problem. However, the paragraph segmentation is much more difficult, and this is due to the highly unstructured texts that can be found in a collection. Thus, we had to use a method that covers almost all the possible paragraph separators that can occur in the texts. The paragraph separators that were implemented so far are: (1) HTML tags, (2) empty lines and (3) paragraph indentations.

Paragraph ordering

Paragraph ordering is performed by a radix sort that involves three different scores: the largest *Same.word.sequence-score*, the largest *Distance-score* and the smallest *Missing.keyword-score*. The definition of these scores is based on the notion of *paragraph-window*. Paragraph-windows are determined by the need to consider separately each match of the same

keyword in the same paragraph. For example, if we have a set of keyword $\{k1, k2, k3, k4\}$ and in a paragraph *k1* and *k2* are matched each twice, whereas *k3* is matched only once, and *k4* is not matched, we are going to have four different windows, defined by the keywords: [*k1-match1, k2-match1, k3*], [*k1-match2, k2-match1, k3*], [*k1-match1, k2-match2, k3*], and [*k1-match2, k2-match2, k3*]. A window comprises all the text between the lowest positioned keyword in the window and the highest positioned keyword in the window. Figure 2 illustrates the four windows for our example.

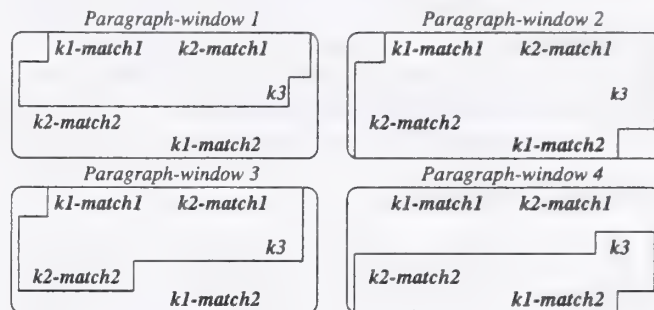


Figure 2: Four windows defined on the same paragraph

For each paragraph window we compute the following scores:

- *Same.word.sequence-score*: computes the number of words from the question that are recognized in the same sequence in the current paragraph-window.
- *Distance-score*: represents the number of words that separate the most distant keywords in the window.
- *Missing.keywords-score*: computes the number of unmatched keywords. This measure is identical for all windows from the same paragraph, but varies for windows from different paragraphs.

The radix sorting takes place across all the window scores for all paragraphs.

Answer Processing

The *Answer Processing* module identifies and extracts the answer from the paragraphs that contain the question keywords. Crucial to the identification of the answer is the recognition of the answer type. Since almost always the answer type is not explicit in the question, nor in the answer, we need to rely on lexico-semantic information, provided by a parser that identifies named entities (e.g. names of people or organizations), monetary units, dates and temporal/locative expressions, as well as products. The recognition of the answer type, through the semantic tag returned by the parser, creates a *candidate answer*. The extraction of the answer and its evaluation are based on a set of heuristics.

The Parser

The parser combines information from broad coverage lexical dictionaries with semantic information that contributes to the identification of the named entities. Since part-of-speech tagging is an intrinsic component of a parser, we have extended Brill's part-of-speech tagger in two ways. First, we have acquired new tagging rules and secondly, we have unified the dictionaries of the tagger with semantic dictionaries derived from the Gazetteers and from WordNet (Miller 1995). In addition to the implementation of grammar rules, we have implemented heuristics capable of recognizing names of persons, organizations, locations, dates, currencies and products. Similar heuristics recognize named entities successfully in IE systems. Having these capabilities proved to be useful for locating the possible answers within a set of candidate paragraphs.

Answer Extraction

The parser enables the recognition of the *answer candidates* in the paragraph. Each expression tagged by the parser with the answer type becomes one of the answer candidates for a paragraph. Similar to the paragraph-windows used in ordering the paragraphs, we establish an *answer-window* for each answer candidate. To evaluate the correctness of each answer candidate, a new evaluation is computed for each answer-window. We use the following scores:

- *Same_word_sequence-score*: it is computed in the same way as for *paragraph-windows*.
 - *Punctuation_sign-score*: is a flag set when the answer candidate is immediately followed by a punctuation sign.
 - *Comma_3_words-score*: measures the number of question words the follow the answer candidate, when the latter is succeeded by a comma. A maximum of three words are sought.
 - *Same_parse_subtree-score*: computes the number of question words found in the same parse sub-tree as the answer candidate.
 - *Same_sentence-score*: computes the number of question words found in the same sentence as the answer candidate.
 - *Matched_keywords-score*: computes the number of keywords matched in the answer-window.
 - *Distance-score*: adds the distances (measured in number of words) between the answer candidate and the other question words in the same window.
- The overall score for a given answer candidate is computed by:

$$\text{Combined-score} = 16 * \text{Same_word_sequence-score} + 16 * \text{Punctuation_sign-score} +$$

$$+ 32 * \text{Comma_3_words-score} + 16 * \text{Same_parse_subtree-score} + 16 * \text{Same_sentence-score} + 16 * \text{Matched_keywords-score} - 4 * \sqrt{\text{Distance} - \text{score}}$$

Currently the combined score represents an unnormalized measure of answer correctness. The answer extraction is performed by choosing the answer candidate with the highest score. Some of the scores approximate very simple abductions. For example, the recognition of keywords or other question words in an apposition determines the *Punctuation_sign-score*, the *Same_parse_subtree-score*, the *Comma_3_words-score* and the *Same_sentence-score* to go up. Moreover, the same sequence score gives higher plausibility to answer candidates that contain in their window sequences of question words that follow the same orders in the question. This score approximates the assumption that concepts are lexicalized in the same manner in the question and in the answer. However, the combined score allows for keywords and question words to be matched in the same order.

Table 3 illustrates some of the scores that were attributed to the candidate answers LASSO has extracted successfully. Currently we compute the same score for both short and long answers, as we analyze in the same way the answer windows.

Question-8	What is the name of the rare neurological disease with symptoms such as : involuntary movements (tics), swearing, and incoherent vocalizations (grunts, shouts, etc)?
Answer (short)	Score: 284.40 <i>who said she has both Tourette's Syndrome and</i>
Question-34	Where is the actress Marion Davies, buried ?
Answer (short)	Score: 142.56 <i>from the fountain inside Hollywood Cemetery</i>
Question-73	Where is the Taj Mahal ?
Answer (long)	Score: 408.00 <i>list of more than 360 cities throughout the world includes the Great Reef in Australia, the Taj Mahal in India, Chartre's Cathedral in France, and Seregenti National Park in Tanzania. The four sites Japan has listed include</i>
Question-176	What is the nationality of Pope John Paul II ?
Answer (long)	Score: 407.06 <i>stabilize the country with its help, the Catholic hierarchy stoutly held out for pluralism, in large part at the urging of Polish-born Pope John Paul II. When the Pope emphatically defended the Solidarity trade union during a 1987 tour of the</i>

Table 3: Examples of LASSO's correctness scores.

Performance evaluation

Table 4 summarizes the scores provided by NIST for our system.

	Percentage of questions in top 5	NIST score
Short answer	68.1%	55.5%
Long answer	77.7%	64.5%

Table 4: Accuracy performance

Another important performance parameter is the *processing time* to answer a question. On the average, the processing time per question is 6 sec., and the time ranges from 1 sec. to 540 sec. There are four main components of the overall time: (1) paragraph search time, (2) paragraph ordering time, (3) answer extraction time, and (4) question processing time. Compared with the rest, the question processing time is negligible. Figure 3 shows the relative percentage (represented on the vertical axis) of the first three components, for the entire range of overall question processing time. The horizontal axis ranks the questions according with their processing time, from the shortest time of 1 sec. to 540 sec.

It can be seen that as the overall time increases, the answer extraction time (including parsing) tends to represent a higher percentage than the rest, meaning that answer extraction is the module where most of the time is spent. This is important as it indicates a bottleneck in the time performance of the system.

Lessons learned

In principle, the problem of finding one or more answers to a question from a very large set of documents can be addressed by creating a context for the question and a knowledge representation of each document and then match the question context against each document representations. This approach is not practical yet since involves advanced techniques in knowledge representation of open text, reasoning, natural language processing, and indexing that currently are beyond the technology state of the art. On the other hand, traditional information retrieval and extraction techniques alone can not be used for question answering due to the need to pinpoint exactly an answer in large collections of open domain texts. Thus, a mixture of natural language processing and information retrieval methods may be the solution for now.

In order to better understand the nature of the QA task and put this into perspective, we offer in Table 5 a taxonomy of question answering systems. It is not sufficient to classify only the types of questions alone,

since for the same question the answer may be easier or more difficult to extract depending on how the answer is phrased in the text. Thus we classify the QA systems, not the questions. We provide a taxonomy based on three criteria that we consider important for building question answering systems: (1) knowledge base, (2) reasoning, and (3) natural language processing and indexing techniques. Knowledge bases and reasoning provide the medium for building question contexts and matching them against text documents. Indexing identifies the text passages where answers may lie, and natural language processing provides a framework for answer extraction.

Out of the 153 questions that our system has answered, 136 belong to Class 1, and 17 to Class 2. Obviously, the questions in Class 2 are more difficult as they require more powerful natural language and reasoning techniques.

As we look for the future, in order to address questions of higher classes we need to handle real-time knowledge acquisition and classification from different domains, coreference, metonymy, special-purpose reasoning, semantic indexing and other advanced techniques.

References

- Sergey Brin and Lawrence Page. The anatomy of a Large-Scale Hypertextual Web Search Engine. In the *Proceedings of the Seventh International World Wide Web Conference*, 1998.
- Chris Buckley, Mandar Mitra, Janet Walz and Claire Cardie. SMART Hight Precision: TREC 7. In the *Proceedings of the Text Retrieval Conference TREC-7*, 1998.
- Paul Cohen, Robert Schrag, Eric Jones, Adam Pease, Albert Lin, Barbara Starr, David Gunning and Murray Burke. The DARPA High Performance Knowledge Bases Project. In *AI Magazine*, Vol 18, No 4, pages 25-49, 1998.
- Paul Cohen, Vinay Chaudhri, Adam Pease and Robert Schrag. Does Prior Knowledge Facilitate the Development of Knowledge-Based Systems? In the *Proceedings of AAAI-99*, 1999.
- Sanda Harabagiu and Dan Moldovan. A Parallel System for Text Inference Using Marker Propagations. *IEEE Transactions in Parallel and Distributed Systems*, Vol 9, no 8, pages 729-748, 1998.
- Sanda Harabagiu and Dan Moldovan. Knowledge Processing on Extended WordNet. In *WordNet: An Electronic Lexical Database and Some of its Applications*, editor Fellbaum, C., MIT Press, Cambridge, MA, 1998.

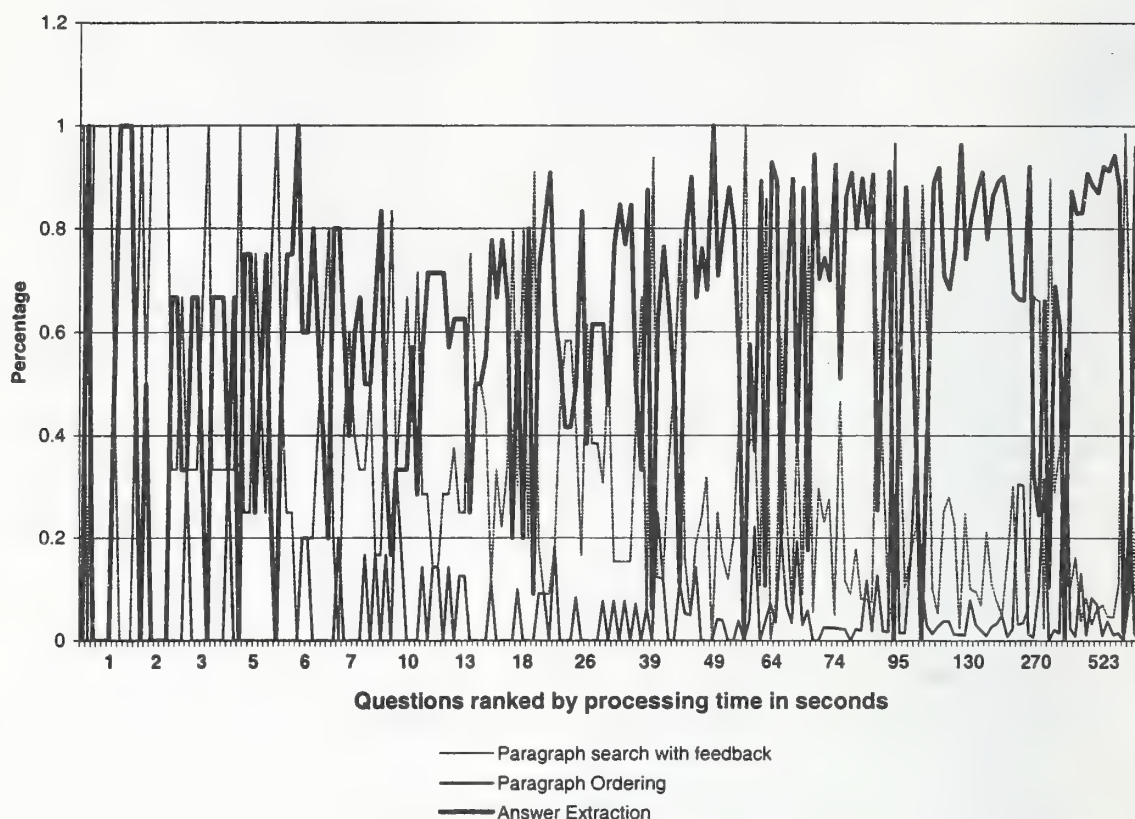


Figure 3: Performance Analysis

Sanda Harabagiu and Steven Maiorano. Finding answers in large collections of texts: paragraph indexing + abductive inference. *Working Notes of the Fall AAAI Symposium on Question Answering*, November 1999.

Marti Hearst. Automated Discovery of WordNet Relations. In *WordNet: An Electronic Lexical Database and Some of its Applications*, editor Fellbaum, C., MIT Press, Cambridge, MA, 1998.

Lynette Hirschman, Marc Light, Eric Breck and John D. Burger. Deep Read: A Reading Comprehension System. In the *Proceedings of the 37th Meeting of the Association for Computational Linguistics (ACL-99)*, pages 325–332, University of Maryland, 1999.

Jerry Hobbs, Mark Stickel, Doug Appelt, and Paul Martin. Interpretation as abduction. *Artificial Intelligence*, 63, pages 69–142, 1993.

Boris Katz. From Sentence Processing to Information

Access on the World Wide Web. *Proceedings of the AAAI Spring Symposium*, pages 77–86, 1997.

Julian Kupiec. MURAX: A Robust Linguistic Approach for Question Answering Using an On-line Encyclopedia. In the *Proceedings of the 16th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-93)*, pages 181–190, Pittsburg, PA, 1993.

G.A. Miller. WordNet: A Lexical Database. *Communication of the ACM*, vol 38: No11, pages 39–41, November 1995.

Dan Moldovan and Rada Mihalcea. A WordNet-based Interface to Internet Search Engines. In *Proceedings of the FLAIRS-98*, pages 275–279, 1998.

Gerard A. Salton. Automatic Text Processing: The transformation, analysis and retrieval of information by computer. Addison-Wesley, 1989.

William A. Woods. *Conceptual Indexing: A Better*

Class	KB	Reasoning	NLP/Indexing	Examples and Comments
1	dictionaries	simple heuristics, pattern matching	complex noun, apposition, simple semantics, keyword indexing	Q33: <i>What is the largest city in Germany?</i> A: .. <i>Berlin, the largest city in Germany..</i> Answer is: simple datum or list of items found verbatim in a sentence or paragraph.
2	ontologies	low level	verb nominalization, semantics, coherence, discourse	Q198: <i>How did Socrates die?</i> A: .. <i>Socrates poisoned himself..</i> Answer is contained in multiple sentences, scattered throughout a document.
3	very large KB	medium level	advanced nlp, semantic indexing	Q: <i>What are the arguments for and against prayer in school?</i> Answer across several texts.
4	Domain KA and Classification, HPKB	high level		Q: <i>Should Fed raise interest rates at their next meeting?</i> Answer across large number of documents, domain specific knowledge acquired automatically.
5	World Knowledge	very high level, special purpose		Q: <i>What should be the US foreign policy in the Balkans now?</i> Answer is a solution to a complex, possible developing scenario.

Table 5: A taxonomy of Question Answering Systems. The degree of complexity increases from Class 1 to Class 5, and it is assumed that the features of a lower class are also available at a higher class.

way to Organize Knowledge. Technical Report of Sun Microsystems Inc., 1997.

Information Extraction Supported Question Answering^{*}

Rohini Srihari and Wei Li

Cymfony Inc.
5500 Main Street
Williamsville, NY 14221, U.S.A.

rohini@cymfony.com, wei@cymfony.com
phone: (716) 565-9114 fax: (716) 565-0308

15 October, 1999

Abstract

This paper discusses the use of our information extraction (IE) system, Textract, in the question-answering (QA) track of the recently held TREC-8 tests. One of our major objectives is to examine how IE can help IR (Information Retrieval) in applications like QA. Our study shows: (i) IE can provide solid support for QA; (ii) low-level IE like Named Entity tagging is often a necessary component in handling most types of questions; (iii) a robust natural language shallow parser provides a structural basis for handling questions; (iv) high-level domain independent IE, i.e. extraction of multiple relationships and general events, is expected to bring about a breakthrough in QA.

1 Introduction

Natural language QA (Question Answering) is an ideal test bed for demonstrating the power of IE (Information Extraction). In our vision, there is a natural co-operation between IE and IR (Information Retrieval).

An important question then is, what type of IE can support IR in QA and how well does it support it? This forms the major topic of this paper. We structure the remaining part of the paper as follows. In Section 2, we first give an overview of the underlying IE technology that Cymfony has been developing. We then present in Section 3 the use of this technology to implement the prototype for the QA Track. In Section 4, we examine question types and discuss their relationship with IE tasks. Finally, in Section 5, we propose a more sophisticated QA system supported by 3 levels of IE.

^{*} This work was supported in part by the following grants from the Air Force, Rome Laboratories: AFRL/IFKRD Phase 2 (Contract No. F30602-98-C-0043) and AFRL/IFKRD Phase 1 (Contract No. F30602-99-C-0102).

2 Overview of Textract IE

The last decade has seen great advances and interest in the area of IE. In the US, the DARPA sponsored Tipster Text Program [Grishman 1997] and the Message Understanding Conferences (MUC) [MUC-7 1998] have been the driving force for developing this technology. In fact, the MUC specifications for various IE tasks have become *de facto* standards in the IE research community. It is therefore necessary to present our IE effort in the context of the MUC program.

MUC divides IE into distinct tasks, namely, NE (Named Entity), TE (Template Element), TR (Template Relation), CO (Co-reference), and ST (Scenario Templates) [Chinchor & Marsh 1998]. Our proposal for three levels of IE is modeled after the MUC standards using MUC-style representation. However, we have modified the MUC IE task definitions in order to make them more useful and more practical.

More precisely, we propose a *hierarchical*, three-level architecture for developing a kernel IE system which is *domain-independent* throughout.

In fact, for level-1 IE, Cymfony has already developed *Textract 1.0*, a state-of-the-art NE tagger [Srihari 1998]. *Textract 1.0* has obtained a score of 91.24% in combined precision and recall (i.e. F-measures), when tested on the MUC-7 dry run data using the MUC-provided scorer. Our tagging speed, approximately 100 MB/hour on a Pentium system, is also comparable to that of the few deployed NE systems, like NetOwl [Krupka & Hausman 1998] and Nymble [Bikel et al 1997].

It is to be noted that, in our definition of NE, we significantly expanded the type of information to be extracted. In addition to all the MUC defined NE types (*person, organization, location, time, date, money and percent*), the following entities are also identified by our existing NE tagger:

- duration, frequency, age
- number, fraction, decimal, ordinal, math equation
- weight, length, temperature, angle, area, capacity, speed, rate
- product, software
- address, email, phone, fax, telex, www
- name (default, i.e. proper name which does not belong to any of the above category)

Sub-type information like *company, government agency, school* (belonging to the type *organization*) and *military person, religious person* (belonging to *person*) are also identified. These new types or sub-types of named entities provide a better foundation for defining multiple relationships between the identified entities and for supporting question answering functionality. For example, the key to a question processor is to identify the **asking point** (*who, what, when, where, etc.*). In many cases, the asking point corresponds to an NE beyond the MUC definition, e.g. the *how-type* questions: *how long* (*duration* or *length* depending on the question context), *how far* (*length*), *how often* (*frequency*), *how old* (*age*), etc. Therefore, an extended NE tagset is helpful for sophisticated IE and QA.

Leve-2 IE, or CE (Correlated Entity), is concerned with extracting *pre-defined* multiple relationships between the entities. This represents a giant step forward from existing deployed IE systems such as *NetOwl, IdentiFinder* [MUC-7 1998] as well as *Cymfony Textract 1.0*, which only output isolated named entities. With CE extraction, salient information is made available to a user since individual, isolated named entities are inter-related. Cymfony has recently implemented a CE

prototype. Consider the *person* entity for example; our CE prototype *Textract 2.0* is capable of extracting the following key relationships:

- **name:** including aliases
- **title:** e.g. Mr.; Prof; etc.
- **subtype:** e.g. MILITARY; RELIGIOUS; etc
- **age:**
- **gender:** e.g. MALE; FEMALE
- **affiliation:**
- **position:**
- **birth_time:**
- **birth_place:**
- **spouse:**
- **parents:**
- **children:**
- **where_from:**
- **address:**
- **phone:**
- **fax:**
- **email:**
- **descriptors:**

As shown, the information in the CE represents a mini-CV of the person. In general, our CE template integrates and greatly enriches the information contained in MUC TE and TR. In terms of relationships, there are only a couple of relationships (*employee_of*, *location_of*) defined in MUC TR.

The final goal of our IE effort is to further extract *open-ended* general events (GE, or level 3 IE) for information like *WHO* did *WHAT* (to *WHOM*) *WHEN* and *WHERE*. By general events, we refer to argument structures centering around verb notions plus the associated information of time and location. GE is dramatically different from the MUC ST task because it is open-ended and domain independent, while ST is pre-defined and highly domain dependent.

Currently, Cymfony is in the stage of research and prototype implementation for this high level IE technology. We show an example of our defined GE extracted from the text (MUC-7 data) below:

Julian Hill, a research chemist whose accidental discovery of a tough, taffylike compound revolutionized everyday life after it proved its worth in warfare and courtship, died on Sunday in Hockessin, Del.

```
[1]  <GE_TEMPLATE> :=  
      PREDICATE:      die  
      ARGUMENT1:      Julian Hill  
      TIME:           Sunday  
      LOCATION:       Hockessin, Del.
```

We show below the overall system architecture for the IE system *Textract* Cymfony has been developing.

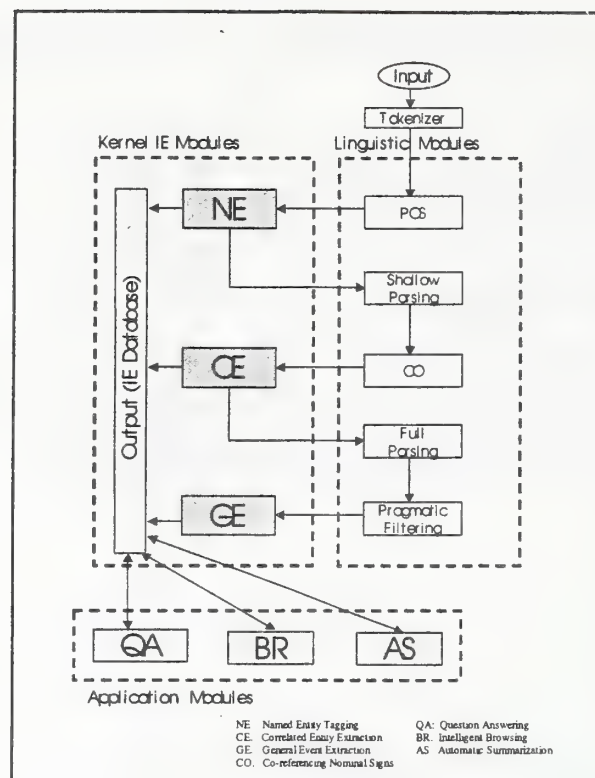


Figure 1: Textract IE System Architecture

As shown, the core of the system consists of 3 kernel IE modules and 6 linguistic modules. These modules remain domain independent. The multi-level linguistic modules serve as an underlying support system for different levels of IE. The IE results are stored in a database which is the basis for IE-related applications like QA, BR (Intelligent Browsing and threading), and AS (Automatic Summarization).¹

It can be shown that each level of IE has immediate application in enhancing the function of IR systems. For example, NE, CE, and GE all support QA. In other words, we do not need to wait until a more sophisticated module is completed before we can try to port our component technology to QA. This is what we have done in TREC QA by employing only NE and a Shallow Parser.

Table 1 gives a concise comparison of the IE task definitions in our architecture and those defined in MUC.

¹ More precisely, the CE technology supports intelligent browsing/threading in the sense that a viewer has access to the collected information about any entity (CE) he is interested in and can jump freely between inter-related entities. The GE results can be used as basis for automatic summarization. When it is adapted to a particular domain, Textract also has a wide variety of application potential (e.g. intelligence, police archive, commercial banking, medical records, classified ads, personalized news abstraction, etc.). However, we believe the key application area is information retrieval.

Our modules	Corresponding MUC definitions	Remark
NE	NE	more NE types defined
CE	TE, TR	more TR types defined in CE
GE	ST	substituting domain dependent ST by our domain independent GE
CO	CO	our CO allows for non-deterministic output

Table 1: Comparison of IE Task Definitions

Our approach to IE consists of a unique blend of machine learning and FST (finite state transducer) rule-based system [Srihari 1998]. By combining machine learning with an FST rule-based system, we are able to exploit the best of both paradigms while overcoming their respective weaknesses.

3 NE-Supported QA

This section presents our approach to QA based on Named Entity tagging. We used our NE tagger *Textract 1.0* for the TREC-8 QA Track and obtained very encouraging results of 66.0% accuracy.

An analysis showed that over 80% out of the 200 questions asked for an NE as a response, e.g. *who* (PERSON), *when* (TIME | DATE), *where* (LOCATION), *how long* (DURATION | LENGTH), *how far* (LENGTH). Therefore, our NE tagger has been proven to be very helpful. For example, in answering questions like "*when did something happen*", the answer should at least contain an NE of time (or date). Questions of the type "*who did this*" usually require an answer containing an NE of person. Of course, the NE of the targeted type is only *necessary* but not complete in answering such questions because NE by nature only extracts isolated individual entities from the text. More sophisticated technology like CE or GE is required in locating the targeted answer. Since the CE prototype *Textract 2.0* was not yet available; we were unable to use this new technology in enhancing our QA performance. Nevertheless, using even simplistic methods like "the nearest NE to the queried key words" or "the NE and its related key words within the same line (or same paragraph, etc.)", NE can help narrow down the targeted text portions which contain potential answers. This is the basic strategy which we employed for our QA prototype.

Would it be possible to achieve the same effect using a traditional search engine without NE support? The answer is largely negative because it is difficult to form a query pertaining to the information about *when* or any NE in general. Although most search engines allow users to form some types of queries using wild cards for pattern matching, they still do not help very much in locating the targeted NE (and associating the NE with something else). This is due to many types of NE (except, perhaps, the NE of *percent*) being expressed in many variable forms in natural language, in fact many more varieties than one usually expects. Even a sophisticated user may not be able to readily come up with good patterns to cover the great number of varieties of NE when forming a query to the system for an answer.²

Figure 2 illustrates the system design of *Textract/QA* Prototype. There are two components for the QA prototype: Question Processor and Text Processor. The Text Matcher module links the two processing results and tries to find answers to the processed question. Matching is based on keywords, plus the NE type (if the question pertains to an entity) and their common location within a same sentence.

² In addition, not all types of NE can be captured by pattern matching effectively. A considerable number of NEs of *person*, *organization* and *location* appear in texts with no obvious surface patterns to be captured. That is exactly the rationale behind our hybrid approach to IE combining pattern matching rules and statistical learning [Srihari 1998].

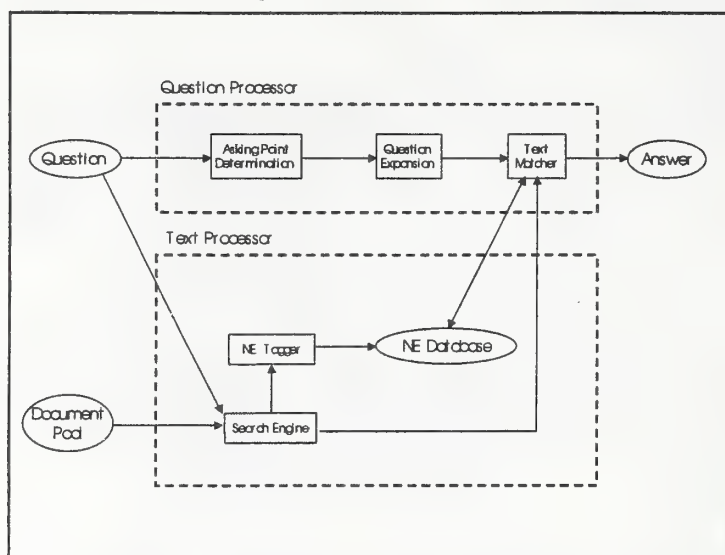


Figure 2: Textract/QA 1.0 Prototype Architecture

Question Processing

The Question Processor consists of two modules: Asking Point Determination and Question Expansion. The results are a list of keywords plus the information for asking point. For example, the question:

[2] *Who won the 1998 Nobel Peace Prize?*

contains the following keywords: *won, 1998, Nobel, Peace, Prize*. The **asking point** *Who* refers to the NE type *person*, which is determined by the module Asking Point Determination. The output before question expansion is a simple 2-feature template as shown below:

[3] *asking_point*: PERSON
 key_word: {won, 1998, Nobel, Peace, Prize}

The following is an example where the asking point does not correspond to any type of NE in our definition:

[4] *Why did the Cultural Revolution occur in China?*

The system then maps it to the following question template:

[5] *asking_point*: REASON
 key_word: {occur, Cultural, Revolution, China}

Basically, the module Asking Point Determination scans the question to search for question words (*wh*-words) and maps them into corresponding NE types/sub-types or pre-defined notions like REASON. We adopt two sets of pattern matching rules for this purpose: (i) structure based pattern matching rules; (ii) simple key word based pattern matching rules (regarded as default rules).

The first set of rules are based on shallow parsing results of the questions using our existing FST based *Textract English Shallow Parser*; the same shallow parser is also used in our IE system, as

shown in Figure 1. This parser identifies basic syntactic constructions like BaseNP (Basic Noun Phrase), BasePP (Basic Prepositional Phrase) and VG (Verb Group).

The following is a sample of the first set of rules:

- [6] Name NP(city|country|company|mountain|river)
--> CITY|COUNTRY|COMPANY|MOUNTAIN|RIVER
- [7] Name NP(person_w) --> PERSON
- [8] Name NP(org_w) --> ORGANIZATION
- [9] Name NP(NOT person_w, NOT org_w) --> NAME

Rule [6] checks the head word of the NP. It covers cases like *VG[Name] NP[a country] that VG[is developing] NP[a magnetic levitation railway system]*. Rule [7] works for cases like *VG[Name] NP[the first private citizen] VG[to fly] PP[in space]* as *citizen* belongs to the word class *person_w*. Rule [9] is a catch-all rule: if the NP is not of class *person* (*person_w*) or organization (*org_w*), then the asking point is a proper name (default NE), often realized in English as a capitalized string of words. Examples include *Name a film that has won the Golden Bear in the Berlin Film Festival*.

We used the following pattern transformations to expand our ruleset:

- (Please) name NP[X]
--> what/which Aux(be) (the name of) NP[X]
--> NP(what/which...X)

In other words, the 4 rules are expanded to 12 rules. For example, Rule [10] below corresponds to Rule [6]; Rule [11] is derived from Rule [7]:

- [10] what/which Aux(be) NP(city|country|company|mountain|river)
--> CITY|COUNTRY|COMPANY|MOUNTAIN|RIVER
- [11] NP(what/which ... person_w) --> PERSON

Rule [10] extracts the asking point from cases like *NP[What] Aux[is] NP[the longest river] PP[in the United States]*. Rule [11] covers the following questions: *NP[What costume designer] VG[decided] that NP[Michael Jackson] VG[should only wear] NP[one glove]*, *NP[Which former Ku Klux Klan member] VG[won] NP[an elected office] PP[in the U.S.]*, *NP[What Nobel laureate] VG[was expelled] PP[from the Philippines] PP[before the conference] PP[on East Timor]*, *NP[What famous communist leader] VG[died] PP[in Mexico City]*, etc.

As seen, shallow parsing helps us to capture a variety of natural language question expressions. However, there are cases where some simple key word based pattern matching would be enough to capture the asking point. That is our second set of rules. These rules are used when the first set of rules have failed to produce results. The following is a sample of such rules:

- [12] who/whom --> PERSON
- [13] when --> TIME/DATE
- [14] where/what place --> LOCATION
- [15] what time (of day) --> TIME

[16]	what day (of the week) -->	DAY
[17]	what/which month -->	MONTH
[18]	what age/how old -->	AGE
[19]	what brand -->	PRODUCT
[20]	what -->	NAME
[21]	how far/tall/high -->	LENGTH
[22]	how large/big/small -->	AREA
[23]	how heavy -->	WEIGHT
[24]	how rich -->	MONEY
[25]	how often -->	FREQUENCY
[26]	how many -->	NUMBER
[27]	how long -->	LENGTH/DURATION
[28]	why/for what -->	REASON

At times, the rule keeps ambiguity unresolved and produces non-deterministic output for the asking point. For example, *how long* maps to NE type *duration* | *length*; the Text Matcher will attempt to find sentences which contain either an NE of *duration* or an NE of *length* and at the same time maximally satisfying the key word based matching constraints. Obviously, this is a naive approach; however, this simple approach is more robust and has proven to be very helpful in our TREC experiments.

The second module in the Question Processor is Question Expansion (corresponding to *query expansion* in IR). Currently, this module is a simple lexical lookup procedure. It attempts to find synonyms to the key verbs (plus their morphological variants) and/or synonyms to the non-NE notion for the asking point. After expansion, the template for the template [3] would be:

[29] *asking_point*: PERSON
 key_word: { won | win | gain | gained | get | got | acquire | acquired |
 obtain | obtained,
 1998, Nobel, Peace, Prize }

The template corresponding to [5] is expanded to the one shown below:

[30] *asking_point*: { because | because of | due to | thanks to | for the reason |
 since | as | for | in order to | in order that | to VB
 }
 key_word: { occur | occurred | happen | happened | take place | took place,
 Cultural, Revolution, China }

The last item in the *asking_point* list attempts to find an infinitive by checking the word *to* followed by a verb (with the part-of-speech tag VB). As we know, infinitive verb phrases are often used in English to explain a reason for some action.

Text Processing

On the text processing side, we first send the question directly to a search engine in order to narrow down the document pool to the first n , say 200, documents for NE processing. We employed the TREC-supplied search engine results from the AT&T engine for this purpose.

The Text Matcher attempts to match the question template with the processed documents for both the asking point and the key words. There is a preliminary ranking algorithm built-into the matcher

in order to find the most probable answers. The primary rank is a count of how many unique keywords are contained within a sentence. The secondary ranking is based on the order that the keywords appear in the sentence compared to their order in the question. The third ranking is based on whether there is an exact match or a variant match for the key verb.

Finally, we implemented a procedure to filter the extracted sentences representing potential answers to a string of 50 bytes or less, as required by the QA Track. This procedure ensures that the output must contain the information, NE or other key words, determined by the asking point.

Cymfony Textract/QA participated in TREC QA in the category of submitting text strings less than 50 bytes. Our accuracy was 66.0%. Considering we have only used NE technology to support QA in this run, 66.0% is a very encouraging result. There is considerable room for performance enhancement. We believe that high level IE like our defined task of CE and GE are more helpful in QA (see 5 for discussion).

4 IE Results as Answers to Questions

We study general question types and how they are related to IE in this section. This suggests our future research direction in terms of using IE for QA. We show that our proposed hierarchical IE system covers a wide range of questions which are not supported by the conventional MUC-defined IE tasks

An IE system performing MUC tasks is helpful in performing the QA function. For example, in answering questions like *where is some company located*, the TR information (*location_of*) provides a ready answer. The identification of different types of NE is also helpful in raising the retrieval hit rate in performing QA. However, there are many more frequently asked questions where MUC IE is of little help. These questions are of two major types, as shown below.

Type I: ask about X's Y (X is an entity, Y is its one aspect/feature, including relationship), e.g.

What is X's address, email and telephone number?

What is X's profession?

Who (or Which organization) is X's employer?

Who is X's wife?

Where is the company X located?

What products does the company X have?

Type II: ask about when, where, who/whom, what of an event, e.g.

When did something happen?

Where did something happen?

Who did that?

Whom did X meet?

What did X give to Y?

To whom did X give Y?

The first are entity-related questions. As any entity is related to its environment in multiple aspects, the couple of TRs defined by MUC only cover a fraction of this type of questions. The design of Correlated Entities is motivated by users' frequent needs to search for various aspects of information (Y) about an entity X, namely Type I questions: what is X's Y. As shown before, we have defined many more relationships which are deemed important for an entity in our CE template. It covers

Type I questions much better than MUC TR. In addition, it is expected that, with implementation of some trivial function on a threaded search, the type of questions which can be answered by our CE system can be extended from X's Y to more general questions of the following type: X1's X2's ...Y? For example, *What is Julian Werver Hill's wife's telephone number?* (equivalent to *What is Polly's telephone number?*) *Where is Werver Hill's affiliated company located?* (equivalent to: *Where is Du Pont Inc. located?*), etc.

The second are event-related questions, but they are about general events. The MUC ST information does not seem to be of help in answering general event questions as it is entirely domain dependent. For example, ST defined by [MUC-7 1998] is specific to the domain of air vehicle launch reports. It is defined to only capture "information about launch vehicle, the payload of that vehicle, the date and site of the launch, and information about the mission type, function and status". This type of extremely domain dependent information does not seem to have a place in general purpose IR/QA. The most powerful QA support, we believe, comes from our General Event design as it can address questions on open-ended domain independent events. Our effort in implementing GE is expected to produce a breakthrough in QA.

In summary, we have pointed out the limitation of the MUC IE standards in QA. We demonstrate that for each level of our domain independent IE, our design serves the purpose of QA much better.

5 Future Work: Multi-level IE Supported QA

Figure 3 illustrates the system architecture for our proposed QA development plan based on three levels of IE support from *Texttract 3.0* (under development).

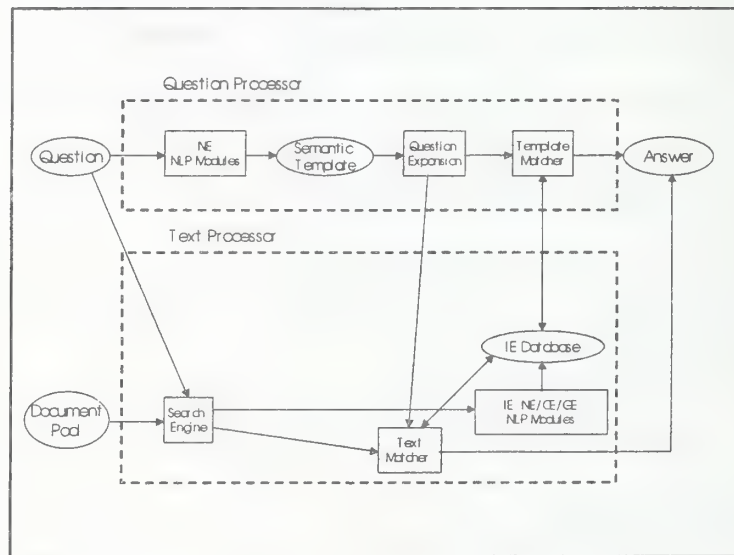


Figure 3: *Texttract/QA 3.0* System Architecture

To tackle the parsing of the questions (Q) and the extraction of an appropriate answer (A) from the free text, there are two sub-systems: (i) Question Processor; (ii) Text Processor. It should be noted that these two sub-systems go through parallel processes and share the same NLP resources until the point of matching and ranking. In fact, questions form a sub-domain of natural language phenomena, following the same syntax. In our design, a question goes through tokenization, POS tagging, NE tagging and shallow parsing and full parsing to generate the semantic template for the question. The only NLP module which does not have a place in question processing is CO since questions are normally related to each other to form a meaningful discourse. This design keeps the

QA system both modular and flexible. Both sub-systems can make use of the same linguistic and IE support.

The merging of question templates and GE templates in the Template Matcher are fairly straightforward. As they both undergo the same NLP processing, the resulting semantic templates are of the same form. Both question templates and GE templates correspond to fairly standard/predictable patterns (the PREDICATE value is open-ended, but the structure remains stable). More precisely, a user can ask questions on general events themselves (*did what*) and/or on the participants of the event (*who, whom, what*) and/or the time and place of events (*when, where*). This addresses by far the most types of general questions of a potential user.

For example, if a user is interested in company acquisition events, he can ask questions like: *Which companies were acquired by Microsoft in 1999? Which companies did Microsoft acquire in 1999?* Our system will then parse these questions into the templates as shown below.

```
[31]  <Q_TEMPLATE> :=  
      PREDICATE:      acquire  
      ARGUMENT1:      Microsoft  
      ARGUMENT2:      WHAT(COMPANY)  
      TIME:           1999
```

If the user wants to know *when* did some acquisition happened, he can ask: *WHEN was Netscape acquired?* Our system will then translate it into the pattern below.

```
[32]  <Q_TEMPLATE> :=  
      PREDICATE:      acquire  
      ARGUMENT1:      WHO  
      ARGUMENT2:      Netscape  
      TIME:           WHEN
```

Note that WHO, WHAT, WHEN above are variables to be instantiated. Such question templates serve as search constraints to filter the events in our extracted GE template database. Because the question templates and the extracted GE template share the same structure (just compare the above question templates [31] [32] with the GE template in [1]), a simple merging operation would accomplish the function of providing the user with exactly the event and its related information he is searching for.

Nevertheless, there are two important questions waiting to be answered: (i) what should be done if a verb used in a question differs from one used in the processed text, even though both verbs convey the same meaning? (ii) what should be done if the question asks something beyond the GE (or CE) information?

The first question occurs when the question is *When was Netscape **acquired*** and the text is *Netscape was **bought** by AOL in 1998*. As discussed previously, although our GE extractor is designed to capture variations of surface structures (e.g. passive expressions) into the semantic structure (*argument structure*) of GE Template, the open ended PREDICATE slot is simply filled by the actual verb used (with inflection removed), *buy* in this case. The same thing happens to question parsing, resulting in *acquire* as the filler of the PREDICATE slot. In order to successfully merge synonyms like *acquire* and *buy*, we have designed a module Question Template Expansion based on ontology. The use of an on-line ontology is conceived in our interface system to automatically produce related patterns based on the question template. For example, from the ; from the template

X acquire/buy Y from Z, another pattern using the trigger word *sell* is automatically produced: *Z sell Y to X*; from the template *X own Y*, a derived template using synonymous trigger word (*have, possess*) is produced: *X have/possess Y* etc. This open-ended question answering feature supported by the GE results and lexical resources of an ontology is the ultimate goal for the QA application of our IE system.

The answer to the second question lies in our 'back-off' model, i.e. the model which we used in the TREC QA Track (presented fully in Figure 2 and included as a part in Figure 3). As this naive QA implementation is based on maximally satisfying the key word constraints plus the NE constraint, it is much more robust. Questions like *why, how* which are beyond the information in our GE (and CE) defined scope³, will still get a list of potential, possibly less accurate, answers.

6 Acknowledgement

We would like to thank our colleagues: William Andrews, Shumin Liu, Lars Nilsson, Cheng Niu, Ray Wen, Hao Xu, and Wenjie Qiao for their assistance in completing the QA task. In addition, we wish to acknowledge Professor Gordon Cormack of the University of Waterloo, Waterloo, Ontario, for providing access to the on-line search engine.

7 References

- Bikel, D.M. *et al.*, 1997. Nymble: a High-Performance Learning Name-finder. *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Morgan Kaufmann Publishers, pp. 194-201.
- Chinchor, N. & Marsh, E. 1998. MUC-7 Information Extraction Task Definition (version 5.1), *Proceedings of MUC-7*
- Grishman, R., 1997. TIPSTER Architecture Design Document Version 2.3. Technical report, DARPA.
- Krupka, G.R. & Hausman, K. 1998. IsoQuest Inc.: Description of the NetOwl (TM) Extractor System as Used for MUC-7, *Proceedings of MUC-7*
- MUC-7, 1998. Proceedings of the Seventh Message Understanding Conference (MUC-7), published on the website <http://www.muc.saic.com/>
- Srihari, R. 1998. A Domain Independent Event Extraction Toolkit, AFRL-IF-RS-TR-1998-152 Final Technical Report, published by Air Force Research Laboratory, Information Directorate, Rome Research Site, New York

³ This is decided by the nature of IE. That is, an effective information extraction system always *ignores* some information and attempts to extract only salient or key information. In contrast, a user may ask questions about any information.

SPOKEN DOCUMENT RETRIEVAL FOR TREC-8 AT CAMBRIDGE UNIVERSITY

S.E. Johnson†, P. Jourlin‡, K. Spärck Jones‡ & P.C. Woodland†

†Cambridge University Engineering Department, Trumpington Street, Cambridge, CB2 1PZ, UK.

Email: {sej28, pcw}@eng.cam.ac.uk

‡Cambridge University Computer Laboratory, Pembroke Street, Cambridge, CB2 3QG, UK.

Email: {pj207, ksjs}@cl.cam.ac.uk

ABSTRACT

This paper presents work done at Cambridge University on the TREC-8 Spoken Document Retrieval (SDR) Track. The 500 hours of broadcast news audio was filtered using an automatic scheme for detecting commercials, and then transcribed using a 2-pass HTK speech recogniser which ran at 13 times real time. The system gave an overall word error rate of 20.5% on the 10 hour scored subset of the corpus, the lowest in the track. Our retrieval engine used an Okapi scheme with traditional stopping and Porter stemming, enhanced with part-of-speech weighting on query terms, a stemmer exceptions list, semantic ‘poset’ indexing, parallel collection frequency weighting, both parallel and traditional blind relevance feedback and document expansion using parallel blind relevance feedback. The final system gave an Average Precision of 55.29% on our transcriptions.

For the case where story boundaries are unknown, a similar retrieval system, without the document expansion, was run on a set of “stories” derived from windowing the transcriptions after removal of commercials. Boundaries were forced at “commercial” or “music” changes and some recombination of temporally close stories was allowed after retrieval. When scoring duplicate story hits and commercials as irrelevant, this system gave an Average Precision of 41.47% on our transcriptions.

The paper also presents results for cross-recogniser experiments using our retrieval strategies on transcriptions from our own first pass output, AT&T, CMU, 2 NIST-run BBN baselines, LIMS1 and Sheffield University, and the relationship between performance and transcription error rate is shown.

1. INTRODUCTION

The TREC-7 Spoken Document Retrieval (SDR) Track showed that successful retrieval of information where the original source of the documents is audio is possible for small collections [4, 5]. The results showed that although retrieval performance degraded when recogniser performance worsened, the fall off was rather gentle and good retrieval can still be achieved on transcriptions with over 100% Processed Term Error Rate [10], corresponding to 66% Word Error Rate (WER) [11]. Further work has shown that various extensions to our retrieval system can increase performance across the whole range of error rates, with

an Average Precision (AveP) of 55.88 obtained on reference transcriptions, 55.08 on our own transcriptions (24.8% WER) and 44.15 on transcriptions from DERA [17] (61.5% WER) on the TREC-7 task [15].

Although by speech recognition standards, the 100 hour test data for TREC-7 represented a large task, the 2866 stories and 23 queries provided only a small collection to test retrieval systems. The conclusions which could be drawn about SDR were therefore limited and a larger collection was needed to confirm the results. The 500 hours of TREC-8 data, with 21,754 stories and 50 queries, represents such a collection and the results presented in this paper show how our methods adapt to a larger task.

An additional feature of our TREC-8 system is that no knowledge about story boundaries is used for recognition, and two retrieval runs are made for each set of transcriptions. For the first run, manual “story” boundaries have been added and commercials have been manually removed (*story-known*) whilst for the second, no such information was used and the retrieval system attempted to find relevant passages in the document collection (*story-unknown*). This led to added challenges in recognition as well as retrieval, with a pre-processing stage being added to remove some data automatically labelled as commercials before recognition began.

This paper firstly describes the TREC-8 SDR tasks and the data used in both development and evaluation of our TREC-8 SDR system. The commercial-detection scheme and the speech recogniser are described in detail in sections 2 and 3 respectively, with the performance of all the sites participating in the cross-recogniser runs also given in the latter. The retrieval engine is then described in section 4, along with a detailed analysis of how the individual retrieval components interacted and affected the overall results. Section 5 focuses on the development of the *story-unknown* system using concatenated TREC-7 data and describes the final evaluation system, giving the results for the TREC-8 task. Cross-recogniser experiments are presented in section 6, where the influence of transcription quality on both the *story-known* and *story-unknown* tasks is investigated. Finally, conclusions are offered in section 7.

1.1. Description of TREC-8 SDR Tasks

The TREC-8 SDR track contains two main tasks. The first, story-known (SK) SDR, is similar to the TREC-7 SDR track, with audio from American broadcast radio and TV news programs provided along with a list of manually-generated *story* (or document) boundaries. Natural language *text* queries, such as “What natural disasters occurred in the world in 1998 causing at least 10 deaths?” are then provided and participating sites must submit a ranked list of potentially relevant documents after running a recognition and retrieval system on the audio data. Real relevance assessments generated by humans are then used to evaluate the ranked list in terms of the standard IR measures of precision and recall. For TREC-8, sites may also run their retrieval system on a “reference” transcription which uses manually-generated closed-caption data, and on other automatically generated transcriptions from NIST (*baselines*) or from other participating sites (*cross-recogniser*).

The second TREC-8 task assumes no knowledge of the story boundaries at both recognition and retrieval time (story-unknown case). The end points of the shows are given as the start time of the first “story” and end time of the last “story” but no other story information, including the location of commercial breaks within the show, can be used. Retrieval then produces a ranked list of shows with time stamps, which are mapped in the scoring procedure to their corresponding story identifiers (IDs). All but the first occurrence of each story is marked irrelevant, as are commercials, before the standard scoring procedure is applied.

For both tasks in TREC-8, the recognition is an *on-line* task, i.e. for any given audio show, only data and information derived from before the day of broadcast can be used. Therefore, unlike for TREC-7, unsupervised adaptation on the test collection can only use data up to and including the current day. Retrieval however is *retrospective* and can use any data up until the last day of the document collection (June 30th 1998). Further details can be found in the TREC-8 specification [6].

1.2. Description of Data

There are two main considerations when describing the data for SDR. Firstly the audio data used for transcription, and secondly the query/relevance set used during retrieval. Table 1 describes the main properties of the former, whilst Table 2 describes the latter, for the *development* (TREC-7) and *evaluation* (TREC-8) data sets.¹

	TREC-7 (dev)	TREC-8 (eval)
Nominal Length of Audio	100 hours	500 hours (SU)
Number of Documents	2,866	21,754 (SK)
Approx. Number of Words	770,000	3,900,000 (SK) 4,700,000 (SU)
Average Doc length	269 words	180 words (SK)

Table 1: Description of data used

¹Only 49 of the 50 queries for TREC-8 were adjudged to have relevant documents within the TREC-8 corpus

	TREC-7 (dev)	TREC-8 (eval)
Number of Queries	23	50
Average Length of Query	14.7 words	13.7 words
Mean # Rel Docs per Query	17.0 docs	36.4 docs (SK)

Table 2: Description of query and relevance sets used

2. AUTOMATIC DETECTION OF COMMERCIALS

To enable both the case of known and unknown story boundary SDR to be investigated, the recognition must be run on all of the 500 hours of audio without using any knowledge of the story boundaries. Since a substantial portion of the data to be transcribed was known to be commercials and thus irrelevant to broadcast news queries, an automatic method of detecting and eliminating such commercials would potentially reduce the number of false matches, thereby increasing the precision of the overall system. Removing commercials early on in processing would also reduce the amount of data that needed to be transcribed and hence speed up the overall recognition system. The first stage of our SDR system was thus a commercial detector designed to eliminate automatically some sections of audio thought to correspond to commercials, whilst retaining all the information-rich news stories.

2.1. Development on TREC-7

The commercial detector was based on finding segments of repeated audio using a direct audio search (described in [12]), making the assumption that (usually) only commercials are repeated. Experiments were performed on the 8.7 hours of TREC-7 SDR data from ABC by searching for segments of similar audio within the data. The results from using 2 sliding window systems with length L and skip S to generate the initial segments are given in Table 3 along with a system which uses the automatically generated wideband segments from our 1997 Hub-4 segmenter [7]. Since the segmentation and commercial detection processes interact, results after both stages are given.

Segments Generation	Cut-Off	Alone		+Segmentation	
		Non-Story	Story	Non-Story	Story
Automatic WB segs	low	31.59%	0.00%	46.27%	1.30%
	medium	36.94%	0.01%	51.62%	1.31%
	high	39.97%	0.24%	54.65%	1.54%
Slide L=10s S=2s	low	59.41%	0.17%	68.35%	1.38%
	medium	62.45%	0.39%	70.27%	1.56%
	high	64.53%	0.57%	71.83%	1.69%
Slide, L=20s S=4s	low	50.30%	0.05%	58.72%	1.33%
	medium	57.75%	0.38%	65.21%	1.60%
	high	63.92%	1.25%	70.41%	2.44%

Table 3: Proportion of story/non-story rejected by direct search on coded audio for TREC-7 ABC data

A low cut-off threshold on the shorter window-length system was chosen to maximise the rejection of commercials whilst keeping the rejection rate of genuine stories below 0.2%. The effect of relabelling segments shorter than a certain smoothing length, T_s , which appeared between two segments labelled as commercials was investigated, with the results given in Table 4.

This shows that smoothing for up to a minute between detected commercial segments increases the performance of the commercial rejection system.

T_s (s)	Non-story Rejection	Story Rejection
0 (none)	59.41%	0.17%
30	62.31%	0.17%
60	70.90%	0.17%
90	73.34%	0.45%

Table 4: Effects of smoothing on TREC-7 ABC data

These general principles were used in the design of the TREC-8 system, but some changes and additions were made to reflect the different nature of the TREC-8 story unknown task: for example, only data broadcast before the current day can be used to identify commercials in TREC-8.

2.2. The TREC-8 System

In a more realistic scenario, the user is not likely to be interested in retrieving information which has been re-broadcast, (i.e. *repeats*) whether it be a commercial or a news story. However, the TREC-8 evaluation set-up meant it was better to retain segments containing news content even if they were repeats, whilst eliminating those repeated segments which correspond to commercials. Safeguards were therefore added to try to reduce the probability of any matching audio which was not a commercial being falsely rejected during the commercial detection stage.

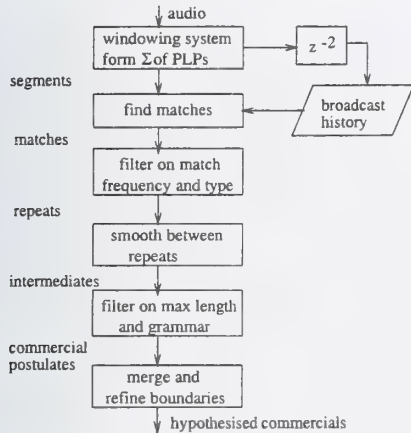


Figure 1: The commercial detection process

A block diagram of the commercial detection process used for the TREC-8 evaluation is given in Figure 1. Audio of the current show was analysed into 5 second windows with a window shift of 1s. Each window was characterised by the covariance matrix of the (wideband) PLP cepstral coefficients as used in the subsequent speech recognition passes. A broadcast history was built up which consisted of the windows for a certain amount of broadcast data (typically 20 hours) from that broadcaster, running up to a few days before the date of the current show. The delay was introduced to reduce the probability of an actual news story occurring in the broadcast history being directly re-broadcast in the current show. The broadcast history was initialised using the January 1998 TDT-2 data and rolled through the TREC-8 SDR evaluation data as the data was processed.

Each segment in the current show was then compared to the segments in the broadcast history. If the arithmetic harmonic sphericity distance [1] between the covariance matrices of the segments was less than a threshold, then the pair was marked as “matching”. Note that a non-zero threshold was necessary, even when looking for identical audio, since there is no guarantee that the sampling and window shifts in each case are synchronous with the audio event in question.

For a segment to be marked as a true repeat, the number of matches between the segment and the broadcast history had to be above a given threshold, to reduce the number of false alarms due to similar, but not identical audio (for example for segments which overlapped by say 80%) matching erroneously. The probability of a re-broadcast story being labelled as a repeat was further reduced by defining the number of different days in the broadcast history which must be involved in the match before the segment was accepted as a repeat.

The merging process was then applied which relabelled as *intermediates* any small gaps which occurred between two segments already labelled as *repeats*. The intermediates were then relabelled as commercials, only if the resulting smoothed “commercial” was less than a critical length, the repeats always being relabelled as commercials. For the CNN shows a show “grammar” (constructed from the CNN TREC-7 data) was used to constrain the locations in the audio that could be labelled as commercials. Due to the limited time resolution of the commercial labelling process, conservative start and end points were also used.

2.3. Results for the TREC-8 System

Since the audio was eliminated at an early stage and could not be recovered later during processing, a very conservative system, COMM-EVAL, which removed 8.4% of the audio, was used for the evaluation. A contrast run, COMM-2, which removed 12.6% of the audio, was later made to see the effect of relaxing the tight constraints on the system. The breakdown of data removed using these systems compared to the manually generated story labels is given in Table 5. Note that these “reference” labels are not an exact reflection of the story/commercial distinction, since a few commercials have been wrongly labelled as stories and some portions of genuine news have not had story labels added and hence are erroneously scored as commercials; however they offer a reasonable indicator of the performance of the commercial detector within the context of this evaluation.

The results show that automatic commercial elimination can be performed very successfully for ABC news shows. More false rejection of stories occurs with CNN data, due to the frequency of short stories, such as sports reports, occurring between commercials. The amount of commercial rejection with the VOA data is low, due mainly to the absence of any VOA broadcast history from before the test data. However, overall the scheme worked well, since 97.8% of the 42.3 hours of data removed by the COMM-EVAL system (and 95.0% of the 63.4 hours removed by the contrast COMM-2 run) were labelled as non-story in the reference.

	Broad.	Non-Stories	Stories	Total
COMM EVAL	CNN	26.19hr=35.7%	2822s=0.46%	27.0hrs=11.0%
	ABC	12.78hr=65.5%	28s=0.02%	12.8hrs=20.5%
	PRI	1.93hr=16.6%	297s=0.10%	2.0hrs= 2.2%
	VOA	0.47hr= 5.0%	132s=0.04%	0.5hrs= 0.5%
	ALL	41.4hrs=36.3%	0.9hrs=0.2%	42.3hrs=8.4%
COMM - 2	CNN	43.26hr=59.0%	10640s=1.73%	46.2hrs=18.9%
	ABC	13.78hr=70.6%	107s=0.07%	13.8hrs=22.1%
	PRI	2.60hr=22.4%	416s=0.14%	2.7hrs= 2.9%
	VOA	0.56hr= 6.0%	208s=0.06%	0.62hrs= 0.6%
	ALL	60.2hrs=52.9%	3.2hrs=0.81%	63.4hrs=12.6%

Table 5: Amount of data rejected during commercial elimination

3. THE TREC-8 HTK BROADCAST NEWS TRANSCRIPTION SYSTEM

After the commercial detection and elimination, the data is automatically segmented and classified by bandwidth and gender. The segmenter initially classifies the data as either wideband (WB) speech, narrowband (NB) speech or pure music/noise, which is discarded. The labelling process uses Gaussian mixture models and incorporates MLLR adaptation. A gender-dependent phone recogniser is then run on the data and the smoothed gender change points and silence points are used in the final segmentation. Putative segments are clustered and successive segments in the same cluster are merged (subject to the segment length remaining between 1 and 30 seconds). The TREC-8 segmenter, which ran in approximately 0.75x real time, included a revised mixture model for music and applied new insertion penalties, but is essentially similar to the system described in [7] with the modifications for faster operation from [18].

Since silence, music and noise are discarded during segmentation, it is interesting to note the interaction between this stage and the commercial elimination phase. The results, given in Table 6, show that the proportion of data discarded by the segmenter decreases from 9.5% to 7.4% if applied after the commercial elimination stage.

	before seg.	after seg.
Original	502.4	454.6
Commercial Elim	460.2	426.0

Table 6: Number of hours of audio retained during processing

The main transcription system used a continuous mixture density, tied-state cross-word context-dependent HMM system based on the CUHTK-Entropic 1998 Hub4 10xRT system [18]. The speech was coded into 13 static cepstral coefficients (including C0) and their first and second derivatives. Cepstral mean normalisation was applied over each segment. After commercial detection and segmentation, a 2-pass recognition system was applied. The initial transcription pass through the data, denoted CUHTK-p1, used gender-independent, bandwidth-specific tri-phone models, with a 60,000 word 4-gram language model to produce a single best hypothesis. The gender of each segment was then labelled by choosing the most likely alignment of this transcription using male and female HMMs. Top-down covariance-based clustering [9] was then applied on a gender and bandwidth

specific basis to all the segments broadcast on a given day and MLLR transforms were generated for these clusters using the first pass transcriptions.

The second pass used the MLLR-adapted gender-dependent tri-phone models with a 108,000 word 3-gram mixture language model to generate lattices from which a one-best output was generated using a 4-gram model. This transcription, denoted CUHTK-s1u, was used for the story-unknown retrieval experiments, whilst the story-known transcription, CUHTK-s1, was simply generated by filtering this output using the known story boundaries. The overall system gave a word error rate of 15.7% on the November 1998 Hub4 evaluation data and 20.5% on the 10-hour scored subset of the TREC-8 evaluation data and runs in about 13xRT on a single processor of a dual processor Pentium III 550MHz running Linux.

The HMMs were trained using 146 hours of broadcast news audio running up to 31st January 1998, supplied by the LDC and used for the 1998 Hub-4 task. The gender-independent wide-band models were generated initially, then narrowband models were created by single pass retraining using a band-limited (125Hz to 3750Hz) analysis. Gender-specific models were generated using a single training iteration to update the mean and mixture weight parameters.

Three fixed backoff word-based language models were trained, from broadcast news text, newspaper texts and acoustic transcriptions, which were all generated using data from before 31st January 1998. The first model was built using 190 million words of broadcast news text, covering 1992-1996 (supplied by the LDC), Nov. 1996 to Jan. 1998 (from the Primary Source Media Broadcast News collection) and Jan. 1998 (from the TDT-2 corpus transcriptions). The LDC also supplied the 70m words from the Washington Post and Los Angeles Times covering 1995 to Jan. 1998, which were used for the newspaper texts model. The third model was built using 1.6m words from the 1997 and 1998 acoustic training transcriptions and 1995 Marketplace transcriptions. Single merged word based models were created which resulted in effectively interpolating the three models, forming a single resultant language model. The final 60k language model had 6.0m bigrams, 14.6m trigrams and 9.4m 4-grams, whilst the 108k model had 6.2m, 14.8m and 9.4m respectively.

3.1. WER Results from Cross-Recogniser Runs

As well as our own transcriptions (CUHTK-s1) we used several alternative sets to assess the effect of error rate on retrieval performance. These came from manually generated closed-captions, both unprocessed (*cc-unproc*) and with some standard text processing of numbers, dates, money amounts and abbreviations (*cc-proc*); two baselines produced by NIST using the BBN Rough 'N' Ready transcription system, (NIST-B1 and NIST-B2), including a fixed and dynamically updated language model respectively; transcriptions from recognisers from LIMSI, Sheffield University, AT&T, and Carnegie Mellon University (CMU); and the output of the first pass of our system (CUHTK-p1).

A 10-hour subset of the TREC-8 (story-known) evaluation data was taken and detailed transcriptions made by the LDC for scoring the recognisers. The results are given in Table 7.

Recogniser	Corr.	Sub.	Del.	Ins.	Err
cc-proc	92.7	2.4	4.9	1.5	8.8
cc-unproc	88.8	4.1	7.1	1.2	12.4
CUHTK-s1	82.4	14.0	3.7	2.9	20.5
LIMSI	82.0	14.6	3.4	3.5	21.5
CUHTK-p1	77.3	18.5	4.2	3.9	26.6
NIST-B2	76.5	17.2	6.2	3.2	26.7
NIST-B1	75.8	17.8	6.4	3.3	27.5
AT&T	75.8	20.4	3.8	5.1	29.3
Sheffield	71.9	22.0	6.1	3.9	32.0
CMU	39.6	28.1	32.3	4.0	64.4

Table 7: WER on 10 hour subset of TREC-8 evaluation data

The results show that the CUHTK-s1 automatic transcriptions are very good, suggesting that the error rate, though some distance from that for the manually-generated closed caption transcriptions, is still low enough not to degrade retrieval performance substantially. It is pleasing to note that the relatively simple CUHTK-p1 system, which uses a smaller vocabulary, has no adaptation and runs in around 3 times real time, gives a reasonably low word error rate.

4. RETRIEVAL SYSTEM

The basic system we used for SK retrieval in TREC-8 is similar to that presented at TREC-7 [11], but the final system also contains several new devices. These include Semantic Poset Indexing (SPI) and Blind Relevance Feedback for query expansion, both on the test collection itself (BRF) and a parallel corpus (PBRF), all of which have been shown to increase performance on the TREC-7 task [14, 15]. A new technique called Parallel Collection Frequency Weighting (PCFW) is also presented along with an implementation of document expansion using the parallel corpus within the framework of the Probabilistic Model.

4.1. System Description

4.1.1. Preprocessing

A term t_i is a set of words or word sequences from queries or documents which are considered to be a unique semantic unit. We call the first set of operations which define the relationship between terms and their components *preprocessing*. The following preprocessing techniques are sequentially applied on all transcriptions and queries before indexing and retrieval.

The words are first made lower case and some punctuation characters are removed. Hyphens and digital numbers were kept even though they do not occur in the ASR-transcribed documents.² Some sequences of words are then mapped to create single compound words. and some single-word mappings are also

²One might think that some hyphens should be removed from the manually transcribed documents (e.g. health-related) whereas others should not (e.g. anti-abortion). Because of a lack of preliminary experiments we decided not to remove any hyphens or digits.

applied to deal with known stemming exceptions and alternative (possibly incorrect) spellings in the manual transcriptions. The list of compound words and mappings was created manually for our TREC-7 SDR system [11]. A set of non-content (stop) words was removed from all documents and queries, with an additional set also being removed from just the queries, e.g. {find, documents, ...}. Abbreviations, (in several forms) are mapped into single words, e.g. [C. N. N. -> cnn].

The use of Porter's well-established stemming algorithm [19] allows several forms of a word to be considered as a unique term, e.g. $t_i(\text{train}) = \{\text{train, training, trainer, trains, ...}\}$. Unlike the mapping techniques, this algorithm is not limited by the use of a fixed thesaurus and therefore every *new* word in a test collection can be associated with its various forms.

4.1.2. Indexing

The *index* (inverted) file contains all the information about a given collection of documents that is needed to compute the document-query scores. For the collection, each term t_i in the term-vocabulary has an associated:

- *collection number* $n(t_i)$: the number of documents which at least one of the components of t_i occurs in.
- list of *term frequencies* $tf(t_i, d)$, which is the number of occurrences of all of the components of t_i in document d .

The index file also contains the number of documents in the collection, N , and the length of each document $dl(d_j)$.

Semantic Poset Indexing (SPI) [14] is used to allow $tf(t_i, d)$ and $n(t_i)$ to take into account some semantic relationships between terms. More specifically, semantic poset structures based on unambiguous noun hyponyms from WordNet [2] and a manually-built geographic locations tree were made. A term occurring in a poset is then redefined as the union of itself and all more specific terms in the poset associated with that term, before the statistics are calculated. For example, the term frequency for a term t_i thus becomes the sum of the frequencies of occurrence of itself and all more specific related terms within a given document.

4.1.3. Retrieval

A part-of-speech (POS) tagger is run over the queries and the weight of each query term t_i is scaled by a factor $pos(t_i)$ using the POS weighting scheme from our TREC-7 system [11]. The score for a document with respect to a given query is then obtained by summing the combined weights, $cw(t_i, d_j)$, for each query term t_i according to the following formulae:

$$cw(t_i, d_j) = \frac{pos(t_i) \cdot (\log N - \log n(t_i)) \cdot tf(t_i, d_j) \cdot (K + 1)}{K \cdot (1 - b + b \cdot ndl(d_j)) + tf(t_i, d_j)}$$

$$n(t_i) = \sum_{d_i \in D} \begin{cases} 0 & tf(t_i, d_i) = 0 \\ 1 & tf(t_i, d_i) > 0 \end{cases}$$

$$dl(d_j) = \sum_{w \in V} tf(w, d_j) \quad ndl(d_j) = \frac{dl(d_j) \cdot N}{\sum_{d \in D} dl(d)}$$

where V is the term vocabulary for the whole document collection D ; and K and b are tuning constants

4.1.4. Blind Relevance Feedback (BRF)

When the documents in the collection are ranked according to a given query, it is possible to expand the query by adding several terms which occur frequently within the top documents but rarely within the whole collection. The T terms which obtain the highest Offer Weight are added to the query. The Offer Weight of a term t_i is :

$$ow(t_i) = r \cdot \log \left[\frac{(r + 0.5)(N - n - R + r + 0.5)}{(n - r + 0.5)(R - r + 0.5)} \right]$$

where R is the number of top documents which are assumed to be relevant; r the number of assumed relevant documents in which at least one component of t_i occurs; n the total number of documents in which at least one component of t_i occurs; and N is the total number of documents in the collection.

4.1.5. Document Parallel Blind Relevance Feedback (DPBRF)

The method of document expansion described within the Vector Model in [20] at TREC-7, can also be used within the probabilistic framework. By considering a document as a *pseudo-query*, it is possible to expand that document using BRF on a parallel collection. For a given document, the 100 terms with the lowest $n(t_i)$ are used as the pseudo-query. BRF is then applied on the parallel collection (with $R = 10$) and the top 400 terms are added to the original document with a term frequency based on their Offer Weight.

4.1.6. Parallel Collection Frequency Weighting (PCFW)

If the test collection is small or contains many transcription errors, the values of $n(t_i)$ may not be sufficiently reliable to use in the prediction of relevance. It is possible to exploit the larger, higher quality parallel collection to obtain better estimates for $n(t_i)$ (and N), to use within the combined weights formula. The collection number, $n(t_i)$, for a given term is therefore replaced by the sum of the collection number for that term on the test corpus and the parallel corpus; with the number of documents, N , being adjusted accordingly.

4.1.7. The Final System

The index file was made as follows:

1. Preprocess & apply SPI to the test collection to give I_t
2. Preprocess & apply SPI to parallel collection to give I_p
3. Perform DPBRF using the pseudo queries from the test collection documents on I_p and add the new terms into the index file I_t .
4. Replace the collection frequency weights in I_t with the PCFWs derived from I_t and I_p and update N accordingly.

The query file was produced by:

1. Preprocess the original natural language request file and attach a POS weight (POSW) to each query term.
2. Perform PBRF using I_p and add the new terms to the query.
3. Perform BRF on I_t and add the new terms to the query.

4.1.8. The Parallel Collection

The parallel collection used in DPBRF, PBRF and PCFW is composed of 51,715 stories extracted from the L.A. Times, Washington Post and New York Times over the period of Jan 1st to June 30th 1998. This contains the TREC-8 SDR test collection period (Feb 1st to June 30th 1998).

4.2. Experiments on TREC-8 SK SDR

The AveP results for our final system on all the sets of transcriptions made available is given in Table 13 in section 6. Here we concentrate on the effect on performance of each aspect of the system on our own CUHTK-s1 transcriptions.

4.2.1. Results on the CUHTK-s1 Transcriptions

It is important to try to understand the contribution of each individual device towards the overall performance of the IR system. Table 8 gives the values of AveP we obtain by progressively decomposing the system.

Lines 1 and 2 show that the addition of all these devices together led to a relative increase in AveP of 23%. Lines 3-5 show that adding just PBRF or BRF individually improve the performance over a system with no blind relevance feedback, but applying PBRF alone gives better results than their combination.

Lines 6-11 show that the improvement due to PCFW is reduced by the use of PBRF. BRF degrades the performance even more when PCFW is present. A similar behaviour can be observed on lines 12-15 for POSW, namely that adding POSW increases performance on the basic system, but degrades when all the other devices are also included. However, this is not the case for DPBRF, as lines 16-17 show that including DPBRF when all other devices are present increases AveP by 5.7% relative.

SPI exhibits a rather different behaviour. It has no significant effect on the baseline system (see lines 18-19), but since the parallel corpus was indexed with SPI, all the devices apart from POSW were affected by the use of this technique. Lines 20 and 21 show that AveP reached 56.72% when SPI was not used and thus SPI actually degraded the performance by 2.5% relative. By comparing lines 20 and 22, we can see that the poor contribution of BRF was due to the inclusion of SPI.

In summary, the inclusion of the techniques discussed increased AveP by 23% relative. Some interaction between the devices was found and it was noted that an AveP of 56.72% could be achieved if SPI had not been included. The corresponding AveP on the processed closed-caption data was 57.66%.

5. THE STORY-UNKNOWN (SU) SYSTEM

For the SU evaluation, no knowledge of the manually-labelled story boundaries can be used either in retrieval or recognition. The system must present a ranked list of show:time stamps, which are mapped to the corresponding story (or commercial) IDs before retrieval performance evaluation, with commercials and duplicates scored as irrelevant.

	SPI	PBRF	PCFW	POSW	PBRF	BRF	AveP	P@30
1	-	-	-	-	-	-	44.96	35.17
2	Y	Y	Y	Y	Y	Y	55.29	40.34
3	Y	Y	Y	Y	-	-	50.63	38.16
4	Y	Y	Y	Y	Y	-	55.69	41.16
5	Y	Y	Y	Y	-	Y	54.27	39.66
6	Y	Y	-	Y	-	-	49.50	37.28
7	Y	Y	Y	Y	-	-	50.63	38.16
8	Y	Y	-	Y	Y	-	55.61	41.43
9	Y	Y	Y	Y	Y	-	55.69	41.16
10	Y	Y	-	Y	Y	Y	55.32	40.41
11	Y	Y	Y	Y	Y	Y	55.29	40.34
12	-	-	-	-	-	-	44.96	35.17
13	-	-	-	Y	-	-	45.90	35.65
14	Y	Y	Y	-	Y	Y	55.49	40.95
15	Y	Y	Y	Y	Y	Y	55.29	40.34
16	Y	-	Y	Y	Y	Y	52.28	38.10
17	Y	Y	Y	Y	Y	Y	55.29	40.34
18	-	-	-	-	-	-	44.96	35.17
19	Y	-	-	-	-	-	44.99	34.90
20	-	Y	Y	Y	Y	Y	56.72	42.72
21	Y	Y	Y	Y	Y	Y	55.29	40.34
22	-	Y	Y	Y	Y	-	55.99	42.31

Table 8: Breakdown of results on the CUHTK-s1 transcriptions showing different combinations of the retrieval techniques

Two main approaches to the SU task exist, the first consists of labelling story boundaries automatically and then running the standard retrieval engine; whilst the second never explicitly finds the story boundaries, but rather locates the relevant passages in the transcriptions and performs some merging of temporally close relevant passages to reduce the possibility of producing multiple hits from the same story source. We investigated one technique from each approach, namely Hearst’s text-tiling [8] for topic boundary detection and a windowing/ recombination system.

For development, the 100 hours of TREC-7 SDR test data was used. This did not exactly model the TREC-8 SU task, since the commercials had already manually been removed from the data, but offered a reasonable basis to compare the different systems. Two methods of scoring were used, the first is the official evaluation scoring procedure, where all instances of a story other than the first one are scored as irrelevant (named *dup-irrel*). The second, by removing all duplicates before scoring, was more lenient and provided an indication of the “best” performance that could be achieved if a perfect merging system (that removed duplicates, but did not re-score or re-order the ranked list) were added after retrieval. This was named *dup-del* and represents a reasonable indication of the potential of any given system.

A simple experiment was conducted to compare a text-tiling system with a windowing system. Text-tiling was originally designed to group paragraphs in long textual reports together and therefore is not ideally suited to the SU-SDR task, since the transcriptions contain no case, sentence or paragraph information. “Pseudo” paragraphs of 10s of speech were made for each show

and the default text-tiling parameters [8] were used along with some additional abbreviations processing, to obtain the “tile” boundaries. Our standard retriever, similar to our TREC-7 system [11], was then used to produce the final ranked list. The windowing system made pseudo-stories of a given length and skip before running the retriever as before. The results are given in Table 9. The windowing system seemed to offer greatest potential and hence the basis of the SU system was chosen to be a sliding window of length 30 seconds and skip 15 seconds.

System	dup-irrel	dup-del	#“Stories”
TREC-7 Story-known	50.3	50.3	2866
Text-tiling	23.2	25.3	4195
Windowing - 120s@60s	28.2	34.0	5226
Windowing - 120s@30s	24.7	35.5	10181
Windowing - 30s@15s	33.9	46.1	18669
Windowing - 30s@10s	27.7	44.0	27890

Table 9: AveP for simple SU systems on the TREC-7 data

The standard retrieval engine was then replaced by a more complicated system, similar to the one described in [14], and *forced-breaks* were added during the windowing to prevent windows being formed over gaps of more than 5 seconds in the audio. Any very short windows (<8 seconds or ≤ 16 words) were removed at this stage. The results are given in Table 10. The increase in performance due to a more sophisticated retrieval engine, which includes SPI and relevance feedback, is clearly shown. Forcing breaks at gaps in the audio did not have much effect on the TREC-7 data (which contained no commercials), but it was hoped that these gaps (generally formed by music/silence removal in the segmentation, or commercial elimination for the TREC-8 system) would offer a good indication of story boundary for the TREC-8 data, and hence should be enforced as hard breaks.

System	dup-irrel	dup-del
Baseline from Table 9	33.9	46.1
Improved Retriever	36.5	51.2
Improved Retriever + forced-breaks	36.0	51.6

Table 10: SU AveP improvements on the TREC-7 data

Post-processing the retrieval output in order to prevent multiple hits of the same story was then examined. Smoothing was added such that for any given query, any stories which were returned as relevant and originated from within a certain time, T_{merge} , in the same broadcast were pooled, with only the highest scoring window being retained. The others were placed in order at the bottom of the ranked list. The results for different values of T_{merge} are given in Table 11.

The results show that the best performance using the TREC-8 evaluation measure (dup-irrel) for the TREC-7 data is obtained with a smoothing time of 105s. This is surprisingly high, but it was thought that the probability that two temporally close windows both being retrieved for a given query but not being

T_{merge}	0	15	30	45	60
dup-irrel	36.0	45.0	45.6	46.2	46.9
dup-del	51.6	51.6	51.1	50.7	50.6
T_{merge}	75	90	105	120	
dup-irrel	47.5	47.8	48.1	48.0	
dup-del	50.6	50.3	50.3	50.0	

Table 11: AveP for different merge times for post-processing on the TREC-7 data

from the same story was quite low. Since the TREC-8 collection contained more data and had a greater proportion of CNN broadcasts, which generally produce shorter stories, the parameter T_{merge} was set to the sub-optimal, but shorter 75s for the TREC-8 evaluation.

Attempts were made to modify the score from the retriever of any window which represented a merged group of windows, before re-ordering during the post-processing phase, but this proved not to be beneficial for the TREC-7 data. Finally hard breaks, as defined by a certain length gap in the audio, were also enforced in the post-processing phase, so that no merging could take place over such a break. The results are given in Table 12 for a T_{merge} of 75 seconds and 120s.

Audio Gap for Boundary	$T_{merge}=75s$		$T_{merge}=120s$	
	dup-irrel	dup-del	dup-irrel	dup-del
100s or ∞	47.51	50.62	48.03	50.03
15s	47.46	50.61	48.05	50.11
10s	47.49	50.63	48.08	50.13
5s	47.46	50.64	48.34	50.45

Table 12: Effect of enforcing hard boundaries in post-processing on TREC-7 data

No real benefit is shown for the TREC-7 data when the smoothing is relatively conservative, but for the case of $T_{merge}=120s$, when the smoothing time is greater than optimal value, the enforcement of boundaries for audio gaps of 5s does increase performance slightly. Since the problem of over-smoothing was thought to be greater for TREC-8 as the commercials had not been manually removed, the enforcement of boundaries at 5s gaps in the audio was maintained.

The final system, summarised in Figure 2, gave an AveP of 41.47 (R-prec=41.98) on our own transcriptions on the TREC-8 task. A more detailed analysis of the SU results for TREC-8 can be found in [13].

6. CROSS-RECOGNISER EXPERIMENTS

Several sets of transcriptions from other participating sites were offered to allow comparisons to be made between retrieval using different recognition systems. The detailed breakdown of the word error rate of these transcriptions is given in Table 7 in section 3.1. The AveP for both the SK and SU runs, along with

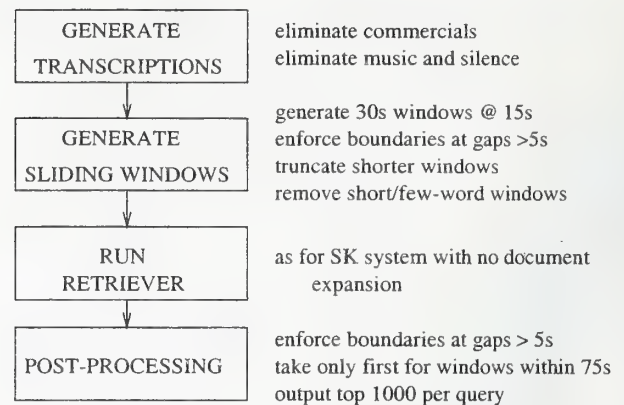


Figure 2: The TREC-8 SU system

the term error rate [10] after stopping and stemming (SSTER) and word error rate (WER) is given in Table 13. The AveP for a benchmark system with no relevance feedback, document expansion or parallel collection frequency weights (BASE) is given as a comparison.³

The term error rate after document expansion (DETER) is also given in Table 13 as a comparison. To calculate this measure, pre-processing, poset mapping and then document expansion are performed on both the reference and hypothesis transcriptions before the standard term error rate is calculated.⁴

Recogniser	Error Rate on 10hr subset			Average Precision		
	WER	SSTER	DETER	SK	BASE	SU
cc-proc	8.8	14.2	30.82	54.93	48.54	—
cc-unproc	12.4	18.0	83.07	52.32	48.93	—
CUHTK-s1	20.5	27.8	45.89	55.29	46.04	41.47
LIMSI	21.5	29.1	47.25	54.12	45.19	40.19
CUHTK-pl	26.6	36.5	56.13	54.51	44.84	41.50
NIST-B2	26.7	35.0	51.56	53.02	43.64	38.70
NIST-B1	27.5	36.1	81.12	49.63	43.25	38.62
AT&T	29.3	38.6	55.73	52.75	43.89	—
Sheffield	32.0	44.7	60.66	52.85	42.47	38.24
CMU	64.4	77.8	103.52	39.36	31.37	—

Table 13: AveP for SK and SU cross-recogniser evaluation conditions with corresponding transcription error rates

Figure 3 shows the relationship between stopped-stemmed term error rates (SSTER) and AveP. Whilst the benchmark (BASE) performance can be predicted reasonably well from SSTER, there is more, seemingly unpredictable, variation for the case of the complete SK system. In particular, the AveP for the NIST-B1

³The unprocessed version of the closed caption transcriptions cc-unproc is not included in all the subsequent analysis since it does not reflect the standard output format

⁴Since there is no guarantee that the terms added to the reference transcriptions during document expansion will be “good” terms for the subsequent retrieval runs, the new “reference” transcriptions may no longer represent the ideal case, but it was hoped that this measure would allow the effects of document expansion to be seen and in particular to show up any major problems which occurred during the document expansion process.

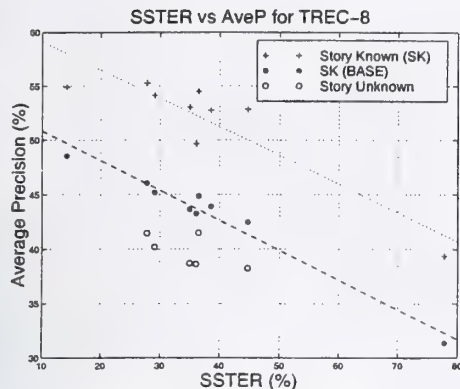


Figure 3: Relationship between AveP and SSTER

and *cc-unproc* runs is much worse than that predicted by the SSTER. However, the DETER for both these cases is unusually high, suggesting the problem for these runs lay in the document expansion process.⁵

It is interesting to note that the best-fit lines for both the complete SK system and the benchmark SK cases are almost parallel, (gradients -0.26 and -0.27 respectively), showing that the inclusion of relevance feedback for query and document expansion and parallel collection frequency weights improves the overall AveP by around 8.5% absolute across the complete range of transcription error rates.

The SU results follow a roughly similar pattern, suggesting that generally transcriptions which work well for the SK case also work well for the SU case. It is pleasing to note that the output from the first pass of our system, CUHTK-p1, does better than might be predicted from its error rate. This is due in part to the reduction in false alarms because of the elimination of commercials in the system. This is confirmed by the results given in Table 14, which show that the AveP on CUHTK-p1 transcriptions would have fallen by 0.5% if the commercial detector had not been used, whereas the performance on LIMSI transcriptions increases by over 0.5% when the detected commercials are filtered out during the post-processing stage (see [13] for more details).

Run	No Commercials removed	COMM-EVAL removed
CUHTK-p1	41.00%	41.50%
LIMSI	40.19%	40.75%

Table 14: Effect on AveP for the SU case when automatic commercial detection is included

6.1. New TERs to Predict Performance

Term Error Rates were introduced in [11] to model the input to the retriever more accurately than the traditional word error rate.

⁵It was found that a disk filling up during the document expansion process for NIST-B1 was responsible for the relatively poor performance for this case. When rectified, the AveP for NIST-B1 was 52.81

If knowledge about the retrieval process itself is known in advance, then the TER can be modified to exploit this information to model the retrieval process more closely and therefore hopefully provide a better predictor of the performance of the final system. An example of this is using SSTER, where the stopping, mapping and stemming processes used in the first stage of indexing the transcriptions, is incorporated into the error rate calculation.

If more information is known about how the scores are generated within the retriever for a given term, then new TERs can be defined which incorporate this information. The generic TER function thus becomes:

$$TER = \frac{\sum_w [f_w(|R(w) - H(w)|)]}{\sum_w [f_w(R(w))]}$$

where f_w is some function which generally depends on the word w , R is the reference and H the hypothesis. This can be seen to reduce to the standard TER when f is the identity function. Some other possibilities for the function f_w which allow the collection frequency weighting (inverse document frequency),⁶ or the combined weights formula to be included directly are:

$$\begin{aligned} f_w(x) &= x/n \\ f_w(x) &= x [\log(N) - \log(n_w)] \\ f_w(x) &= \frac{x [\log(N) - \log(n_w)] (K + 1)}{x + K[1 - b + b \text{ndf}]} \end{aligned} \quad (1)$$

where N , K , b , n and ndf have the same meaning as in section 4.1.3. It is also possible to include the frequency of each term in the *query* as a scale factor within f_w if the queries are known, but this makes the score query-dependent, which may be undesirable, and care must be taken in defining the query terms if relevance feedback is used for query expansion.

The TERs using (1), including stopping, stemming, mapping, posets, document expansion and parallel collection frequency weights within the combined weighting formula are given in Table 15. Unfortunately these numbers do not appear to offer a better predictor for our AveP results. This may be because the words added to the “reference” during document expansion may not be the best in terms of retrieval performance, or that only the query terms themselves should be taken into account, or simply the overall performance on the entire 500 hour collection cannot be predicted well using the scored 10 hour subset.

Rec. Error	HTK 55.51	cc-proc 37.93	HTK-p1 67.86	LIMSI 57.20	NIST-B2 62.46
Rec. Error	Sheff 72.80	AT&T 66.79	cc-unproc 104.75	NIST-B1 91.90	CMU 121.68

Table 15: Term error rate modelling stopping, stemming, mapping, posets, document expansion and PCFW with combined weighting on the scored 10 hour subset

⁶Another method of modifying the TER to model retrieval weighting more closely can be found in [20]

7. CONCLUSIONS

This paper has described the systems developed at Cambridge University for the 1999 TREC-8 SDR story known and story unknown evaluations.

A new method of automatically detecting commercials has been shown to work well, with 97.8% of the 42.3 hours of data automatically labelled as commercials being marked as non-story information by humans. By automatically eliminating these "commercials" at an early stage, the computational effort required during speech recognition was reduced by 8.4% and the Average Precision for the story unknown task was increased by 1.2% relative.

Two HTK-based transcription systems were made. The first ran in 3 times real time and gave a word error rate (WER) of 26.6% on the scored 10 hour subset of the data. The second ran at 13 times real time and included a second pass with a 108k vocabulary and speaker adaptation, giving a WER of 20.5%, the lowest in the track by a statistically significant margin.

Several extensions to our retriever have been described and shown to increase Average Precision on our best transcriptions for the story-known case by 23% relative, giving a final value of 55.29%. These included semantic poset indexing, blind relevance feedback, parallel blind relevance feedback for both query and document expansion and parallel collection frequency weighting.

The system developed for the case where story boundaries were not known included automatic detection and elimination of commercials, windowing using the segmentation information, retrieval using all the strategies developed for the story-known case except document expansion, and post-filtering to recombine multiple hits from the same story. The final system gave an average precision of 41.5% on both sets of our transcriptions.

Finally, experiments were described using other transcriptions and the relationship between transcription error rate and performance was investigated. The results from TREC-7 showing that the degradation of performance with increasing error rate was fairly gentle were confirmed on this significantly larger data set.

Acknowledgements

This work is in part funded by an EPSRC grant reference GR/L49611. Thanks to Tony Robinson for the initial idea that repetitions of audio could help to indicate the presence of commercials.

8. REFERENCES

- [1] F. Bimbot & L. Mathan *Text-Free Speaker Recognition using an Arithmetic Harmonic Sphericity Measure*. Proc. Eurospeech '93, Vol. 1. pp. 169-172, 1993
- [2] C. Fellbaum *WordNet: An Electronic Lexical Database* MIT Press, ISBN 0-262-06197-X, 1998
- [3] M.F.G. Gales & P.C. Woodland *Mean and Variance Adaptation Within the MLLR Framework* Computer Speech & Language, Vol. 10 pp. 249-264, 1996
- [4] J.S. Garofolo, E.M. Voorhees, C.G.P. Auzanne, V.M. Stanford & B.A. Lund *The 1998 TREC-7 Spoken Document Retrieval Track Overview and Results* Proc. TREC-7, pp. 79-89, 1999
- [5] J.S. Garofolo, E.M. Voorhees, C.G.P. Auzanne & V.M. Stanford *Spoken Document Retrieval: 1998 Evaluation and Investigation of New Metrics* Proc. ESCA Workshop on Extracting Information from Spoken Audio, pp. 1-7, 1999
- [6] J.S. Garofolo, C.G.P. Auzanne, E.M. Voorhees, K. Spärck Jones *1999 TREC-8 Spoken Document Retrieval (SDR) Track Evaluation Specification*
http://www.nist.gov/speech/sdr99/doc/sdr99_spec.htm
- [7] T. Hain, S.E. Johnson, A. Tuerk, P.C. Woodland & S.J. Young *Segment Generation and Clustering in the HTK Broadcast News Transcription System* Proc. DARPA Broadcast News Transcription and Understanding Workshop, pp. 133-137, 1998
- [8] M.A. Hearst *TextTiling: Segmenting text into multi-paragraph subtopic passages* Computational Linguistics, Vol. 23. pp. 33-64, 1997 (Source code <http://elib.cs.berkeley.edu/src/texttiles/>)
- [9] S.E. Johnson & P.C. Woodland *Speaker Clustering Using Direct Maximisation of the MLLR-Adapted Likelihood* Proc. ICSLP 98, Vol. 5 pp. 1775-1778, 1998
- [10] S.E. Johnson, P. Jourlin, G.L. Moore, K. Spärck Jones & P.C. Woodland *The Cambridge University Spoken Document Retrieval System* Proc. ICASSP'99, Vol. 1, pp. 49-52, 1999
- [11] S.E. Johnson, P. Jourlin, G.L. Moore, K. Spärck Jones & P.C. Woodland *Spoken Document Retrieval for TREC-7 at Cambridge University* Proc. TREC-7, pp. 191-200, 1999
- [12] S.E. Johnson, P.C. Woodland *A Method for Direct Audio Search with Applications to Indexing and Retrieval* To appear in ICASSP 2000
- [13] S.E. Johnson, P. Jourlin, G.L. Moore, K. Spärck Jones & P.C. Woodland *Audio Indexing and Retrieval of Complete Broadcast News Shows* To appear in RIAO 2000
- [14] P. Jourlin, S.E. Johnson, K. Spärck Jones & P.C. Woodland *General Query Expansion Techniques for Spoken Document Retrieval* Proc. ESCA Workshop on Extracting Information from Spoken Audio, pp. 8-13, 1999
- [15] P. Jourlin, S.E. Johnson, K. Spärck Jones & P.C. Woodland *Improving Retrieval on Imperfect Speech Transcriptions* Proc. SIGIR '99 pp. 283-284, 1999
- [16] C.J. Leggetter & P.C. Woodland *Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models* Computer Speech & Language, Vol. 9 pp. 171-185, 1995
- [17] P. Nowell *Experiments in Spoken Document Retrieval at DERA-SRU* Proc. TREC-7, pp. 353-362, 1999
- [18] J.J. Odell, P.C. Woodland & T. Hain *The CUHTK-Entropic 10xRT Broadcast News Transcription System* Proc. 1999 DARPA Broadcast News Workshop, pp. 271-275, 1999
- [19] M.F. Porter *An Algorithm for Suffix Stripping* Program, 14 pp. 130-137, 1980
- [20] A. Singhal & F. Pereira *Document Expansion for Speech Retrieval* Proc. SIGIR '99, pp. 34-41, 1999

Oracle at Trec8: A Lexical Approach¹

Kavi Mahesh, Jacquelynn Kud, Paul Dixon

Oracle Corporation
500 Oracle Parkway M/S 40p10
Redwood Shores, CA 94065 USA
trec@us.oracle.com

Abstract

Oracle's system for Trec8 was the *interMedia Text* retrieval engine integrated with the *Oracle8i* database and SQL query language. *interMedia Text* supports a novel theme-based document retrieval capability using an extensive lexical knowledge base. Trec8 queries constructed by extracting themes from topic titles and descriptions were manually refined. Queries were simple and intuitive. Oracle's results demonstrate that knowledge-based retrieval is a viable and scalable solution for information retrieval and that statistical training and tuning on the document collection is unnecessary for good performance in Trec.

1. Introduction

Oracle's approach to Trec8 is a novel theme-based retrieval method implemented on top of traditional boolean techniques. Theme-based retrieval enables querying for documents that are *about* a certain theme or concept. The set of themes from a document together define what a document is about. Themes are extracted from documents and queries by parsing them using an extensive lexicon together with a knowledge base of concepts and relations. High precision is achieved by a disambiguation and ranking technique called *theme proving* whereby a knowledge base relation is verified in the lexical and semantic context of the text in a document.

Oracle's solution, known as *interMedia Text* is a part of the Oracle database with a standard SQL interface for creating indexes and issuing queries. Database integration and SQL interface make the retrieval engine highly scalable and easy to use. For Trec8 (ad hoc, manual), Oracle used its *interMedia Text* product without any modifications and with absolutely no training or tuning on the Trec document collection. As such, Oracle's solution is expected to deliver equally good quality and performance in other topic domains or with other document collections.

2. Theme-based Document Retrieval

interMedia Text builds an inverted index of a document collection that is either stored or referenced (via file paths, URLs, or procedures) in a database column. In addition, it also adds a hierarchy of ranked themes extracted from each document to the inverted index. Information retrieval queries are part of SQL SELECT statements and hence can be easily combined with other structured queries (e.g., to query document metadata). Queries return hitlists with relative scores in the range of 1..100 that can be used for ranking the hits.

1. Product availability: *Oracle8i interMedia Text Release 2 (8.1.6)* is currently available in the market. The product is also available for download for Oracle Technology Network members at <http://technet.oracle.com>

interMedia Text supports theme or concept-based retrieval using the ABOUT operator which extracts themes from free text queries to match against themes in the inverted index. Theme queries are inherently hierarchical in nature. For example, our query from Topic 436 (railway accidents) included 'ABOUT(rail transportation)' which is in effect equivalent to expanding the term to a large number of narrower concepts and their synonyms using the knowledge base. Figure 1 shows some of the terms in this hierarchy. However, theme queries are not expanded; instead document themes are indexed hierarchically so that documents containing any of the narrower terms would be indexed against the theme rail transportation. Hence, theme queries yield high recall numbers and run much faster than an expanded query. Recall is also increased by recognizing noun phrases and by converting synonyms and alternate forms to canonical forms.

CATEGORY: rail transportation

Conrail, Incorporated
Amtrak
Association of American Railroads
Norfolk Southern Corporation
Southern Pacific
Union Pacific Corporation
automatic train control, automatic train controls, atc system, atc systems
brakepersons, brakeman, brakemen, brakewoman, brakewomen, brakeperson
broad gauges
cog railroads
cowcatchers
gandy dancers
monorails
rail stations, railroad stations, railroad terminals, railway stations
railroad tunnels
railroads, railways
siderodromomaniacs
stationmasters
streetcars
tramways

....

SUBCATEGORY: trains

bullet trains, commuter trains, express trains, goods trains, passenger trains, shuttle trains
center dump ballast cars, chair cars, dining cars, drop-bottom dump cars, log cars, mail cars, observation cars
railcars, railway cars, sleeping cars, tank cars, tender cars

....

SUBCATEGORY: subways

London Underground
Metropolitan Railway
New York City Subway
BART - Bay Area Rapid Transit
Washington D. C. Metro

....

Fig. 1: Knowledge base content for rail transportation. A sample of over 600 terms under this category is shown.

Moreover, not every document that contains the narrower terms (e.g., “trains”) are indexed against rail transportation due to a process called *theme proving* which increases precision. High precision is achieved by proving document themes in the lexical and semantic context of the text in the document. Two themes prove each other if they are closely connected in the knowledge base either hierarchically or through cross references. This eliminates many bad hits arising from word sense ambiguities.

There is also no need for users to formulate extremely long queries or develop a thesaurus for a topic such as rail transportation. ABOUT queries on such terms are effective since interMedia Text has an extensive knowledge base that was built with a focus on information retrieval.

Theme queries are simple and intuitive. For example, shown below is our query for Topic 414 (Cuba, sugar, exports) which obtained the best recall and an average precision well above that topic’s median:

```
‘ABOUT(cuba) * 3, ABOUT(sugar) * 5, ABOUT(Russia) * 2,  
(ABOUT(export) or ABOUT(import))’
```

ABOUT queries can be combined with boolean operators and term weighting as shown above. The ‘,’ is an abbreviation for the ACCUMULATE operator.¹ ABOUT queries can also be combined easily with conventional text queries. For example, our query for Topic 450 (King Hussein, peace) was:

```
‘(King Hussein=Hussayn=Husayn=Husyan * 5,  
ABOUT(peace talks) * 3, peace * 2, ABOUT(Israel))  
MINUS (Iraq or ABOUT(Iraq))’
```

2.1. Lexical Knowledge Base

Theme extraction depends heavily on a good quality lexical knowledge base. Oracle’s knowledge base, built in-house, contains over 200,000 concepts from very broad domains classified into 2000 major categories. These categories are organized hierarchically under six top terms: business and economics, science and technology, geography, government and military, social environment, and abstract ideas and concepts. Concept classification, choice of categories, and the hierarchical organization are all carefully designed for their usefulness in information retrieval rather than ontological purity, with a special focus on avoiding problems of semantic ambiguity. For example, unlike other linguistic or AI ontologies [Bateman et al, 1990; Bouaud et al, 1995; Guarino, 1997; Lenat and Guha, 1990; Mahesh, 1996; Miller, 1990], the top terms are not terms low in content such as “noun”, “thing”, “event”, or “role”.

The hierarchy is organized by a mix of relations such as taxonomic, partitive and other relations, the choice being driven by the salience of the relation rather than uniformity across the ontology. For example, SQL - Structured Query Language is under databases because of its strong association with databases although it is really a programming language. Other relations are encoded as cross references to avoid any ambiguity in parent-child relations. Cross references across the hierarchies play a role in theme proving to enable disambiguation and better ranking of themes.

Each concept is mapped to one or more words or phrases that are in their canonical form. Each canonical form has several alternate forms such as inflectional variants, abbreviations, acronyms, alternate spellings, and synonyms. Canonical form mappings are carefully controlled in the knowledge base to give higher

1. The ACCUMULATE operator is ideal for Trec queries. It behaves like OR except it assigns higher ranks to documents that match more terms and terms with higher term weights.

accuracy than a stemming approach. The total vocabulary of 450,000 terms includes a large number of proper names with an extensive coverage of names of places, organizations, and people.

Each entry has a large set of flags indicating its lexical attributes such as its part of speech as well as several attributes that indicate its information content and the degree of its importance for information retrieval. The latter set of flags has been carefully assigned to each entry in the lexicon. This is a key source of information used for recognizing important themes in a document, thereby improving theme ranking.

2.2. Theme Extraction

Known phrases are recognized using a greedy algorithm. Unknown words and proper name phrases are recognized and treated as themes. Words and phrases are normalized to their canonical forms to avoid having to expand query terms. Every normalized term is a potential theme for the document.

Precision at n hits is improved by theme ranking. Themes are assigned initial ranks based on their flags in the knowledge base, as well as several factors derived from the structure of the document and the frequency of the theme in the document. If a theme maps to a concept in the knowledge base and it is successfully proven in the document context, all of its ancestors in the knowledge base hierarchy are also included as themes of the document.

Ranks of parent themes are determined by their distance to the descendant themes, as well as by the accumulated strength of all descendent themes in the theme hierarchies present in the document. Theme ranks are also adjusted based on the amount of evidence from related themes that proved the theme. Documents in a hitlist are ranked by combining the scores of document themes that match query themes and any user assigned term weights in the query.

Theme proving is an efficient partial solution to the general problem of word sense ambiguity. It essentially verifies if the single predominant sense encoded in the knowledge base is valid in the current document context. Two themes prove each other if they are closely connected in the knowledge base either hierarchically or through cross references. If a theme is not proven, its parent themes are not included in the index. This eliminates many bad hits arising from word sense ambiguities.

For example, document LA082890-0005 is about a plane crash and it mentions that the plane was used to train pilots. This document is not relevant for Topic 436 (railway accidents) in spite of the words “accident” and “train”. This is included in the NIST *qrels* (i.e., it was retrieved as one of the top 100 hits for this topic by at least one system). *interMedia Text* did not retrieve this document at all although the word “train” is under rail transportation in its knowledge base. Theme proving determined that there was no other evidence for rail transportation in this document and hence rejected the railway sense of “train”.

3. Trec8 Run

Oracle used its Oracle8i *interMedia Text* Release 2 (8.1.6) product without any modifications. Only the public interface of the product was used to produce the Trec8 run. Oracle’s run was produced on a Sun Sparc Ultra 1 workstation running the Solaris 2.6 operating system.

3.1. Filtering and Indexing

Documents were extracted into separate files, one per file. An attempt was made to filter the documents to extract just the text bodies using the DTDs specified for each collection. However, since about 4000 docu-

ments did not strictly conform to the DTDs and were empty upon filtering, our final approach was to remove the LA Times index terms that were to be excluded as stated in the Trec guidelines [Trec8 Guidelines] and simply filter out all SGML tags. File names and paths were entered into an Oracle database table. A combined index (i.e., a theme index plus an inverted text index) was built on the collection thus:

```
CREATE INDEX trec7_index ON trec7_docs(doc_text) INDEXTYPE IS ConText;
```

3.2. Training and Tuning

There was absolutely no training or system tuning on the Trec document collection. The lexical knowledge base was not modified in any way. No new stoplist was created. In fact, the index was built before the Trec8 topics were available.

3.3. Query Formulation

All queries were strictly text and theme queries. No other structured criteria were used in the SQL queries. The first set of queries was constructed automatically using the natural language theme query ABOUT operator:¹

```
SELECT doc_id FROM trec7_docs WHERE  
CONTAINS (doc_text, 'ABOUT(<title>), ABOUT(<description>)', 1)>0  
ORDER BY score(1) DESCENDING;
```

These initial free text queries were converted to preliminary boolean queries using interMedia Text's Explain feature. In effect, this amounted to extracting themes from the title and description. This involves recognizing phrases, discarding function words and useless words, and normalizing the words and phrases to their canonical forms using the knowledge base. All of the themes were accumulated in the resulting internal query (using the ACCUMULATE operator).

3.4. Manual Query Refinement

25 topics each were assigned to two members of our team. Each member browsed the top few hits from each topic to identify good and bad query terms. interMedia Text's Theme Highlighting feature was used to obtain an HTML rendering of the document with highlights. The feature highlights not only the words and phrases that were normalized to a theme in the query, but also all of its related and supporting terms (identified during the theme proving operation). This enabled the user to quickly grasp why a particular document was retrieved for the query. Figure 2 shows a relevant document for Topic 436 (railway accidents) that is highlighted for the theme query 'ABOUT(rail transportation)'.

1. interMedia Text assigns scores in the range 1..100. However, in the final submission, each hit had a unique weight = (1000 - rank) instead of the score in the 1..100 range. This was done to avoid any unpredictability in the results since the trec_eval program uses an unstable sorting routine (qsort), i.e., it does not preserve the ranks of hits with equal scores when computing precision and recall numbers.

NATION IN BRIEF;

MARYLAND;

FINAL CLAIM SETTLED IN FATAL **TRAIN** WRECK

From Times Staff and Wire Reports

Conrail agreed to pay \$5.5 million in the last of about 370 court claims stemming from a fiery **Amtrak-Conrail train** wreck that killed 16 people near Baltimore in 1987.

Attorneys for Susan Schweitzer, 45, of New York, said the settlement came just before selection of a jury to hear the civil damage trial in Baltimore. The crash occurred Jan. 4, 1987, when three **Conrail locomotives** ran through a switch and into the path of an **Amtrak train**. **Conrail** engineer Ricky L. Gates later admitted that he and **brakeman** Edward Cromwell had smoked marijuana just before the crash.

Fig. 2: *HTML rendering of document LA032790-0024 (Topic 436) with theme highlighting for 'ABOUT(rail transportation)'.*

The two team members made relevance judgments of the top few hits as they browsed the documents. The resulting *qrels* was a small fraction of the NIST *qrels* but was very useful to track improvements as the users refined queries based on their findings.

Unlike NIST, our team made a three-valued judgment: YES, NO, or MAYBE. The intended semantics of MAYBE was “try to make sure this document is retrieved, but ideally it should rank below the YESs” (similar to “iffy” in [Cormack et al, 1998]). This greatly improved productivity since the members did not have to scrutinize borderline documents to make a binary YES/NO decision.

It turned out that the three-valued judgments were meaningful. Our agreement with NIST *qrels* was 86% considering only YES/NO and was 77% when MAYBEs were treated as YES. Also, NIST *qrels* included 86% of our MAYBEs. 42% of our MAYBEs were considered relevant by NIST.

3.5. Increasing Recall

Queries were manually adjusted to increase recall for some topics by substituting a concept one level above the original theme in the knowledge base. For example, for Topic 449 (antibiotics ineffectiveness), the original query had “antibiotics”. However, not all drugs that are commonly considered antibiotics were under this concept. The knowledge base had a finer classification of these drugs and it appeared better to go with the parent concept of antibiotics, antimicrobials, and antiparasitics. The knowledge base was also used to manually expand queries by adding related terms and sibling concepts for some topics. Some terms that were only present in the narrative part of topics were also added. For example, in Topic 418 (quilts, income), “quilting books” and “quilting classes” were added from the narrative.

It may be noted that the above use of the knowledge base does not require access to the internal knowledge base structure. interMedia Text provides a Hierarchical Query Feedback feature which allows users to browse broader, narrower, and related terms for a query term in the knowledge base.

Also, many queries initially did not return 1000 hits. Although the team members found no particular reason to return more hits, the queries were expanded to return at least 1000 hits since there is no penalty in Trec for returning irrelevant documents after all the relevant ones.

3.6. Increasing Precision

Precision and relevance ranking were improved on some topics by tightening the query in several ways. Boolean operators AND and NOT were used instead of ACCUMULATE. Ambiguous terms, such as “drugs” in topic 449 (antibiotics ineffectiveness), were either deleted from the query or term weights were adjusted to lower their impact on relevance ranking. The MINUS operator (which reduces the score of a hit if the specified term is present) was also used to push borderline hits lower in the hitlist. When several terms were being ACCUMULATED, equivalent terms were grouped under an equivalence (=) or OR clause to prevent dilution of term weights.

Some of these refinements are illustrated by our query for Topic 418 (quilts, income) which obtained the best recall and the best average precision:

‘quilted * 6, {quilting books} * 5, {quilting classes} * 4, quilt=quilts * 0.1,
(ABOUT(quilts) AND (ABOUT(income) or ABOUT(fundraising))) * 2’

4. Trec8 Results

Oracle’s results are summarized in Table 1. Eight topics had a 100% recall and 18 topics had over 90% recall. 5 topics had the best recall at 1000 and the best average precision. Topic 447 (Stirling engine) obtained 100% recall and 100% average precision. Table 2 shows the distribution of our per topic results against the median numbers for that topic.

Table 1: Oracle’s Trec8 Results

Trec8 measure	Orcl99man
Recall at 1000	71.57% (3384/4728)
Average Precision	41.30%
R-precision	43.57%
Initial precision (at recall 0.0)	92.79%
“Final” precision (at recall 1.0)	07.91%

Table 2: Distribution of Oracle’s Per Topic Results in Relation to the Median Numbers

	best	>= median	< median	worst
Recall at 1000 hits	16	19	15	0
Average precision	7	26	17	0

5. Discussion

Oracle used its manual relevance judgments to refine queries. However, relevance ranking of hits was done entirely by the interMedia Text system and was not in any way influenced by the relevance judgments. Other participants have interpreted the guidelines for the manual ad hoc track to allow a reordering of hits using manual relevance judgments. Oracle's results with such reordering are shown below for comparison.

```
Queryid (Num):      50
Total number of documents over all queries
  Retrieved:      50000
  Relevant:        4728
  Rel_ret:       3384
Interpolated Recall - Precision Averages:
  at 0.00        0.9604
  at 0.10        0.8704
  at 0.20        0.7715
  at 0.30        0.6723
  at 0.40        0.5536
  at 0.50        0.4775
  at 0.60        0.3598
  at 0.70        0.2830
  at 0.80        0.2128
  at 0.90        0.1339
  at 1.00        0.0769
Average precision (non-interpolated) for all rel docs(averaged over que-
ries)
                                0.4740
Precision:
  At    5 docs:    0.8640
  At   10 docs:    0.7940
  At   15 docs:    0.7453
  At   20 docs:    0.6930
  At   30 docs:    0.6207
  At  100 docs:    0.3740
  At  200 docs:    0.2389
  At  500 docs:    0.1214
  At 1000 docs:    0.0677
R-Precision (precision after R (= num_rel for a query) docs retrieved):
  Exact:          0.4800
```

Our results would have been better and queries simpler if we could have extended the knowledge base for Trec8 topics. Although Trec8 guidelines [Trec8 Guidelines] say that lexicons may not be extended after seeing the topics, interMedia Text provides an extensibility feature that allows any ordinary user to extend its knowledge base by importing a thesaurus in an ISO-2788 like format [ISO2788, 1986]. Yet, Oracle decided not to extend its knowledge base for Trec8.

An analysis of the few topics where our results were poor (i.e., both recall and average precision significantly below the median) indicated that in most cases, the major cause was our team members had interpreted these topics differently than the NIST evaluators. These were also the topics for which our team members had doubts and did not expect good results. For example, Topic 405 (cosmic events) depended on what was considered an event. The *qrels* indicate that the discovery of some galaxies was an event. Our team member's interpretation was that it had to be an explosion, collision, and so on in order to qualify as a cosmic event. The query had been refined to rank documents about the discovery of celestial objects

lower in the hitlist. As a result, such documents fell beyond the first 1000 in the hitlist.

It is well known that the Trec document collection and retrieval results are overanalyzed [e.g., Banks et al, 1999]. We believe that many systems, especially statistical ones are too well trained on this collection. We have demonstrated that our system can do well with no training on the collection. We are eagerly looking forward to the new document collection to be used in Trec9.

6. Conclusions

Theme queries are simple and intuitive. They are easy to explain in symbolic terms to see exactly why a document was retrieved and assigned a certain rank. Theme-based retrieval requires no training on a document set. It works well across domains and document collections. Database integration makes it scalable and easy to use. The ability to combine theme queries with boolean keyword queries and structured SQL queries offers a flexible and complete solution to information retrieval.

Knowledge-based hierarchical expansion and term normalization improve recall. Theme ranking and proving improve precision--both overall and average precision. Theme-based retrieval also finds a significant number of unique relevant documents (our submission found about 110 unique relevant documents). The ability for ordinary users to customize and extend the knowledge base makes it easy to tailor the retrieval engine to particular applications.

Oracle's results demonstrate that knowledge-based retrieval is a viable and highly scalable solution for information retrieval and that statistical training and tuning on a document collection is unnecessary for good performance.

Acknowledgments

The authors wish to thank Shamim Alpha and Garrett Kaminaga for their ideas and contributions.

References

- [Banks et al, 1999] Banks, D., Over, P., and Zhang, N. Blind Men and Elephants: Six Approaches to TREC data, *Information Retrieval* 1, 7-34, 1999.
- [Bateman et al, 1990] Bateman, J. A., Kasper, R. T., Moore J. D., and Whitney, R. A. A general organization of knowledge for NLP: The PENMAN upper model. Technical Report, USC/Information Sciences Institute, 1990.
- [Bouaud et al, 1995] Bouaud, J., Bachimont, B., Charlet, J., and Zweigenbaum, P. Methodological Principles for Structuring an "Ontology". In *Proc. Workshop on Basic Ontological Issues in Knowledge Sharing, International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada, August 1995.
- [Cormack et al, 1998] Cormack, G. V., Palmer, C. R., Biesbrouck, M. V., and Clarke, C. L. A. Deriving very short queries for high precision and recall (MultiText experiments for TREC-7). In *NIST Special Publication 500-242: The Seventh Text REtrieval Conference (TREC 7)*. National Institute of Standards and Technology, 1998.
- [Guarino, 1997] Guarino, N. Some Ontological Principles for a Unified Top-Level Ontology. *Spring Symposium on Ontological Engineering*, AAAI Tech Report SS-97-06, American Association for Artificial Intelligence, March 1997.

[ISO2788, 1986] International Standard ISO 2788: Documentation - Guidelines for the establishment and development of monolingual thesauri, Second edition - 1986-11-15, International Organization for Standardization.

[Lenat and Guha, 1990] Lenat, D. B. and Guha, R. V. Building Large Knowledge-Based Systems. Reading, MA: Addison-Wesley, 1990.

[Mahesh, 1996] Mahesh, K. Ontology Development for Machine Translation: Ideology and Methodology. Technical Report MCCS-96-292, Computing Research Laboratory, New Mexico State University, 1996.

[Miller, 1990] Miller, G. WordNet: An on-line lexical database. *International Journal of Lexicography* 3(4) (Special Issue).

[Trec8 Guidelines] http://trec.nist.gov/act_part/guidelines.html

TREC-8 Ad-Hoc, Query and Filtering Track Experiments using PIRCS

K.L. Kwok, L. Grunfeld and M. Chan

Computer Science Department, Queens College, CUNY
Flushing, NY 11367

Abstract

In TREC-8, we participated in automatic ad-hoc retrieval as well as the query and filtering tracks. The theme of our participation is 'retrieval lists combination', and the technique is applied throughout our experiments to various degree. It is pointed out that our PIRCS system may be considered as a combination of probabilistic retrieval model and a language model approach. For ad-hoc, three types of experiments were done with short, medium and long queries as before. General approach is similar to TREC-7, but combination of retrieval lists from different query types were used to boost effectiveness. For query track, we submitted one short-query set, and performed retrieval for twenty one natural language query variants. For filtering track, experiments for adaptive, batch filtering, and routing were performed. For adaptive, historical selected document list was used to train profile term weights and dynamically vary retrieval status value (rsv) threshold for deciding document selection during the course of filtering. For batch filtering, Financial Times FT92 data was used to define 6 retrieval profiles whose results were combined based on coefficients trained via a genetic algorithm. Logistic regression transforms rsv's to probabilities. Routing was similarly done with additional training data obtained from non-FT collections and two additional profiles were defined and combined

1. Introduction

We continue to use our PIRCS system for investigation. A theme that we emphasize this year is 'retrieval combination'. Given an information need different query formulation or different search algorithms may retrieve quite different document sets. Combining their retrieval status values (RSV) may reinforce common relevant ones and lead to new ranking that is more effective than the original separate sets. The idea has been in existence in IR practice and literature, and proposed by many people for many years. We employ it to various degree as a way to refine our various experiments.

The basic PIRCS system is a combination of two retrieval algorithms: document-focused and query-focused. In Section 2, we point out that document-focused weighting is similar to weighting based on a language model approach.

In addition to combination, two strategies for ad-hoc are: 2-stage retrieval and collection enrichment as done in TREC-7. Both strategies have been found to work more often than not for queries of different lengths. Ad-hoc retrievals are discussed in Section 3.

In query track, we use our system with 21 variants of topics numbered #51-100 to retrieve on Disk 1, and some observations of the results are given in Section 4.

In the filtering track, adaptive filtering was done by using accumulated selected documents to help set RSV thresholds for future document selection. Batch filtering makes use of FT92 known data to help train multiple variant profiles and their (near) optimal combination coefficients. These were used to simulate final filtering on FT93 & 94 without adaptation. In routing additional profiles, coefficients and training data were used to produce ranked outputs. Adaptive filtering is described in Section 5, batch filtering in Section 6, and routing retrieval in Section 7. Section 8 contains our conclusions.

2. PIRCS Weighting and Language Model

Given a query q to retrieve documents d from a collection, our basic PIRCS system itself is a combination of two retrieval algorithms producing a document-focused and a query-focused RSV's for each document with a mixing parameter α . Thus (see [Kwok95] for greater details):

$$RSV(q,d) = \alpha * RSV_d + (1 - \alpha) * RSV_q \quad (1)$$

with

$$RSV_d = \sum_k S(qtf_k/L_q) * w_{dk} \quad (2a)$$

$$w_{dk} = \log [tf_k/(L_d - tf_k) * (Nw - L_d - F_k + tf_k)/(F_k - tf_k)] \quad (2b)$$

and

$$RSVq = \sum_k S(tf_k/L_d) * w_{qk} \quad (3a)$$

$$w_{qk} = \log [qtf_k/(L_q - qtf_k) * (Nw - F_k)/F_k] \quad (3b)$$

where tf_k , qtf_k are the frequency of term k in d and q respectively, $L_d = \sum_k tf_k$, $L_q = \sum_k qtf_k$ are the lengths of d and q , S is a sigmoid-like function, $F_k = \sum_{all\ doc} tf_k$ is the collection frequency of term k , and $Nw = \sum_k F_k$ is the number of tokens used in the collection.

Our approach considers a document (or query) as constituted of conceptual components approximated as single terms and self-relevant to the document (query) itself, and we work in a universe consisting of document components rather than documents. Because of the self-relevance assumption, every query (document) therefore has a relevant and irrelevant set even when no relevant judgment has been made, we are able to bootstrap and provide probabilistic weights to our terms at the initial retrieval stage. Because we work with conceptual components, repeat term usage and item lengths are accounted for, enabling us to remove the binary assumption restriction. The weighting of Eqn. 3b is the familiar probabilistic query term weights but in the component environment. Eqn. 2b is for document-focused retrieval and the form of the weighting, after taking the approximation $Nw \gg$ all other frequencies, turns out to be very similar to those used by [HiKr98] via a language model approach, but with a different smoothing coefficient.

Thus, our PIRCS system may also be viewed as a combination of the probabilistic retrieval model and a simple language model. For many of the past TREC experiments, our system has been demonstrated to provide superior effectiveness, and last year it was observed that PIRCS is one of few automatic systems that provides many unique relevant documents in the judgment pool [VoHa98]. We believe this is because our system is unique among participants in that it is a combination of two different models.

3 Ad-Hoc Retrieval

The target collection for ad hoc retrieval is from Disks 4&5, consisting of articles from Financial Times, Federal Register, Foreign Broadcast Information Service and the LA Times, some 2 GB of text in over 1/2 million documents. These are similar to TREC-7 and we used last year's processed data unchanged.

TREC-8 topics are described in several sections: title, description and narrative. This year, the official ad-hoc runs should make use of the title and description sections

only. We call this run *pir9Attd*. It is a combination of retrieval lists from *pir9At0* (title only) and *pir9Atd0* (title and description). We consider this group to be short to medium queries. In addition we have two more submitted runs called *pir9Aa1* and *pir9Aatd*, the former has queries making use of all sections, and the latter combines an all section run with *pir9Atd0*. The title, title+description, and all section queries have on average 2.54, 6.14 and 12.8 unique terms respectively after stemming and stopword removal.

Results for short and medium queries are discussed in Section 3.1 and long queries in Section 3.2.

3.1 Short and Medium Queries

We follow our TREC-7 approach to short query retrieval by using five methods successively to produce a final retrieval list. These five methods [KwCh98] are: 1) average within-document term frequency to weight short query terms (avtf query term weighting); 2) variable high frequency Zipfian threshold dependent on query size; 3) collection enrichment to improve initial stage output relevant density; 4) enhancing term variety in raw queries by adding highly associated terms from initial retrieval based on mutual information measure; and 5) using retrieved document local term statistics to improve weighting conditioned on irrelevancy in final retrieval.

	Query Type					
	submit`d		submit`d		official	
	pir9At0 value	% inc	pir9Atd0 value	% inc	pir9Attd value	% inc
Relv.Ret	3299	0	3272	-1	3342	1
Avg Prec	.3063	0	.3022	-1	.3207	5
P@10	.4800	0	.4920	3	.5080	6
P@20	.4410	0	.4290	-3	.4450	1
P@30	.4027	0	.3807	-5	.4033	0
R.Prec	.3326	0	.3301	-1	.3441	3

Table 3.1a: Automatic Ad Hoc Results for 50 Short and Medium Queries

	Query Type											
	unsubmit`d		official		submit`d		<corrected runs >					
	pir9Aa0 value	% inc	pir9Aatd value	% inc	pir9Aa1 value	% inc	pir9Aa1* value	% inc	pir9Aatd* value	% inc		
Relv. Ret	3344	0	3382	1	2751	-18	3350	0	3383	1		
Avg Prec	.3241	0	.3303	2	.2624	-19	.3249	0	.3322	3		
P@10	.4940	0	.5120	4	.4500	-9	.5040	2	.5140	4		
P@20	.4530	0	.4600	2	.3860	-15	.4470	-1	.4610	2		
P@30	.4080	0	.4120	1	.3327	-18	.4033	-1	.4107	1		
R.Prec	.3441	0	.3512	2	.3065	-11	.3421	-1	.3515	2		

Table 3.1b: Automatic Ad-Hoc Results for 50 Long Queries

For collection enrichment, we form a miscellaneous collection by retrieving the top 200 documents from the sub-collections AP1-3, WSJ1-2, FB6 and using the title form of the queries. This miscellaneous collection is used to enrich the top-ranked set of the initial stage retrieval from the target collection. This year we modified the method slightly by limiting the number of external documents for feedback to a maximum so as not to overwhelm expansion based on documents from the target. It helps for TREC-7 but is slightly worse for TREC-8. We also employ combination of retrieval lists to help improve effectiveness; coefficients of combination are learnt from TREC5 to 7 results.

Results and Discussion

Our TREC-8 results for short and medium queries are summarized in Table 3.1, and their nomenclature has been described in the Introduction. The title only (t0: mean av. prec. 0.3063) and title+description (td0: mean av. prec. 0.3022) runs are very close, with a slight edge to the former. This year there are several highly specific topics with words like 'osteoporosis' #404, 'Schengen agreement' #410, 'killer bee attacks' #430, 'supercritical fluids' #444. They are better with the title alone than adding the description. Title only has 26 queries with better average precision, 19 worse and 5 equal to title+description. However, for retrieved relevants at 1000, the numbers are reversed: 14:15:21. Longer queries generally tend to get better recall as was also found in our previous TRECs. Best result is obtained by combining their retrieval lists (ttd) giving improvements of about 5% over (t0), and is our designated official run. It also has a relevant retrieved at 1000 documents of 3342 which is about 70.7% of the pooled documents that have been judged relevant (4728).

Comparisons with the all-sites median average-precision, precision at 100 and 1000 documents are given in Table 3.2. Our runs are well above median. For example, the official combination run (ttd) has average precision better than median in 43 instances with 3 queries achieving the best, and are worse than median in 7 cases. For title only (t0), the number of queries with precision better, equal or worse than median are: 35:4:11. Out of the 35, 11 have the best values. This year the title only and title+description medians are evaluated separately.

Year to Year Comparison

This year's very short (title only) ad-hoc effectiveness is much better than TREC-7, and deserves some discussion since both years use the same collection. Within our site for example, last year's MAP (mean average precision) was 0.2427 for the title only run and 0.3063 this year. The MAP difference of 0.0636 (over 20% improvement) does not seem explainable by parameter adjustments alone. The reason is because the topics for this year appears much easier. If we use a value of $AP \Rightarrow 0.5$ as an indicator of easy topics, then there are 8 this year and only 4 in TREC-7. These 8 and their key terms are: 403 (osteoporosis), 410 (Schengen), 415 (golden triangle), 420 (carbon monoxide), 423 (milosovic), 430 (killer bee), 441 (Lyme), and 444 (supercritical fluid). The AP sum of this 8 totals 5.6507. The sum of last year's 4 easy topics plus the next 4 top totaled 4.5988. This estimated difference amortized over 50 queries contributes .0210 to the MAP difference, or 33% of the observed increase. Thus, in year to year comparison, as we already noted in TREC-7, topic hardness can play a substantial rule.

3.2 Long Queries

Long queries can use all sections of a topic. Our official long query run is pir9Aatd, which is a combination of the title+description only run pir9Atd0 and another that uses all wordings of a topic pir9Aa0 (un-submitted). In addition, we submitted another run called pir9Aa1, which is the basic pir9Aa0 with phrase-reranking added. Unfortunately, an error was committed during the phrasing operation. Each topic content was first POS-tagged and each sentence was broken down into noun phrases. A choice can be made to keep only the noun phrases or to keep the residual entries such as verbs, adverbs as well. The wrong choice of keeping more than noun phrases was made. This leads to erroneous re-ranking of the retrieved documents and bad results for pir9Aa1. After results were known, we re-do the pir9Aa1 run correctly (now called pir9Aa1*), and these are tabulated in Table 1b. By some fortune, our pir9Aatd combination run was done using the uncorrupted pir9Aa0, and it gives very good results. Had we combined pirc9Atd0 with the corrected pir9Aa1* run, the result would be slightly better as shown under the pir9Aatd* column in Table 3.1b.

It is seen from Table 3.1b that phrase re-ranking in pir9Aa1* (with mean average precision of 0.3249) does not do much to pir9Aa0 (0.3241). Combining the all section run with the title+description run can lead to about 2-3% improvements over the components. When compared to results from all sites, Table 3.2, our official

official official

	pir9Aa0 > = <	pir9Atd0 > = <	pir9Atd > = <	pir9Aa1 > = <	pir9Aatd > = <
AvgPrec	35,11 4 11	34,11 1 15	43,3 0 7	24,1 0 26	37,6 3 10
RR@100	34,18 8 8	36,5 4 10	41,9 3 6	22,7 6 22	36,11 4 10
RR@1K	33,21 11 6	38,14 7 5	40,14 5 5	18,9 12 20	39,18 7 4

Table 3.2: Ad-Hoc Results: Comparing All Submitted Runs with Median

long query run has 40 queries equal or above the median with 6 of them being best, and 10 queries worse than median.

4. Query Track

The purpose of the query track is to explore how query variants of the same topical concepts may affect retrieval results. Topics used are 51-100 and retrieval was done on Disk 1. The data consists of variations in average non-interpolated precision for three dimensions T, Q and R. T represents different topics. Q denotes different query compositions for each topic (total of 23 query variants, 21 of which are natural language type and one is our pirla. Two more are weighted query types which we did not analyze. 'pir' denotes our system. The '1' in '1a' means very short version; longer versions like a sentence are denoted by '2' or '3'. R means different retrieval algorithms (specifically 8 of them like INQ, Sab, etc; one of which pir is our PIRCS system). Readers are advised to refer to the query track report for a description of these queries and retrieval algorithms.

For comparison we will use the average non-interpolated precision. We first try to see which query type does best for each topic using our PIRCS retrieval engine by noting the best retrieval within each topic (i.e. R=pir, for each T, find best Q). It shows query type Sab1c performs best 9 out of 50 times, and a group of 6 other query types (INQ1c, INQ2e, INQ3d, Sab1b, Sab3a and pirla) perform best 4 out of 50. Others have less. It seems the Sab1c query formulation agrees well with our engine. When we evaluate the average precision over all topics for each of the 21 query type using our engine (i.e. R=pir, for each Q average over all T), our pirla formulation returns the best performance at 0.3030. Putting this in perspective, the title section of the TREC original topics gives an average precision of 0.2973. If the title, description and narrative sections are used to produce long queries, the average precision is 0.3330.

When the data is averaged over 21 query types for all 50 topics (i.e. for each R, average over all Q and T), we can see how each retrieval algorithm performs. It seems that our pir method returns an average precision of 0.2458, practically the same as Sabe's 0.2459.

When the data is averaged over 21 query types and all 8 algorithms (i.e. for each T, average over all Q and R), one may get some idea of how hard each topic is for retrieval. Average precision varies from 0.6527 (topic 70) to 0.0131 (topic 74). Unlike the current ad-hoc experiments, there are no topics with highly specific terms like 'osteoporosis' or 'Schengen agreement' that can return precision values of 0.8 or higher. Topics 58 (0.5640) and 59 (precision 0.0988) seem to represent one

easy and one hard topic, and we choose them to have a closer look.

Since we cannot run other retrieval algorithms, we focus on the results for topic 58 and 59 returned by our pir engine. For topic 58, out of 21 query types, only 5 have average precision of less than 0.6884, showing that it is an easy topic. The reason these 5 do not do well is because the words 'rail strike(s)' were not used in their formulation. Instead, 'railroad strike', 'strikes .. against

	query	Av.Prec. Initial	Av.Prec. Final
1	rail strikes (has 'railstrike') - NIST title	0.6872	0.7537
2	rail strike reports (") - INQ1a	0.6858	0.7497
3	rail strikes, walkouts (") - pirla	0.6413	0.7364
4	strikes by rail (no 'railstrike')	0.6755	0.7347
5	NIST long query (has 'railstrike')	0.5393	0.7173
6	railstrikes	0.2133	0.6915
7	rail walkouts	0.3957	0.6754
8	railway strikes	0.2704	0.3908
9	railroad strikes	0.3165	0.2946
10	Line1 (.5) combine Line7 (.5)		0.7414
11	(.7) (.3)		0.7534
12	(.8) (.2)		0.7556

Table 4.1a: Topic #58 - Average Precision for Different Query Variants

railroads', or 'labor relations .. in transportation industry' are used.

Table 4.1a shows some deliberate variations of wordings for Topic 58. For human understanding, 'rail strike', 'railroad strike', and 'railway strike' seem synonymous; yet for retrieval the latter ones are much worse (Lines 1, 8 & 9, average precision of .7537, .3908, .2946 respectively). The juxtaposition of 'rail strike' also contribute an additional 2-word phrase 'railstrike' in our system, but its effect is small (Lines 1 & 4) and does not account for those large differences. Lines 2 & 3 shows the idiosyncrasy of IR: one would expect 'walkouts' to add more content and focus to 'rail strikes', yet it is worse than adding 'reports'. Paragraph size query is not as good as the two words in this case as shown in Line 5. Lines 7, 8 & 9 show that the term 'rail' is critical for this query concerning strikes. But, how does one know during query formulation? The document frequency of rail, railroad and railway are: 3108, 3902, 1516 and do not seem able to indicate the usefulness of one or the other. Perhaps one may say that 'rail strike(s)' is the normal description of this concept.

In the spirit of our theme, we try combining queries of Line 1 & 7 giving results in Lines 10, 11 & 12. With the right coefficient, it can surpass the best of its constituents. Even choosing a coefficient of 0.5 is not

bad at 0.7414 average precision, better than using the UNION of such words in a single query shown in Line 3.

Topic 59 is one kind of hard topic, with some of its variant results shown in Table 4.1b. Out of 21 query types, only 8 achieve precision above 0.1 (best is Line 1 of 0.3420) according to our pir engine. It asks for a very

	query	Av.Prec Initial	Av.Prec Final
1	storm deaths - Sab1b	0.2255	0.3420
2	what damaging weather events have caused deaths - INQ2b	0.1205	0.3206
3	storm fatalities	0.0869	0.3147
4	has violent rain storms caused many deaths - INQ3d	0.3173	0.3099
5	deaths caused by storms s/as typhoons, hurricanes, tornados- Sab3a	0.3098	0.2744
6	weather deaths, injuries - pirla	0.0699	0.2718
7	NIST long topic	0.0157	0.0060
8	weather related fatalities - NIST title	0.0218	0.0267
9	Line1 (.5) combine Line6 (.5)		0.4253
10	(.7) (.3)		0.4197
11	(.8) (.2)		0.3930
12	Line1 UNION Line6	0.2709	0.4035

Table 4.1b: Topic #59 - Average Precision for Different Query Variants

general concept 'weather related fatalities'. Two simple words 'storm deaths' is best for this retrieval (Line 1), while 'storm fatalities' (Line 3) is also very good. It turns out that 'fatalities' without 'storm' is a bad choice. Queries using 'weather' with 'fatalities' all return miserable results like (Line 8). 'deaths' seem to be a more effective choice (Line 2 & 6) although it is difficult to see why one is better than the other at query formulation. 'Weather related' is very general, and it would seem spelling out the more common occurring specific cases such as: hurricanes, tornadoes, floods, rain etc. may be more useful. Line 4 did just that but surprisingly it was only good (0.2744) and not the best. The word 'storm' seems to capture the concept 'weather related bad things' well as it is less polysynous than 'weather'. Combination of retrieval lists (Lines 9, 10 & 11) or combining terms in one query i.e. longer query (Line 12) can boost effectiveness substantially even for this hard query.

The study of these two queries only shows that the choice of words for retrieval is crucial for good results. How to make a good choice is not at all clear.

5 Adaptive Filtering Track

This year's adaptive filtering task makes use of topics

#350-400 to select documents from the FT (Financial Times) collection from 1992 to 1994 in date order. Adaptive filtering is difficult. A possible approach is to use a two step strategy: at start when little knowledge is known, a simple adaptive threshold-adjustment and profile re-weighting method is used. After sufficient relevant data is available, train and expand profiles carefully and do filtering without adaptation like in batch filtering. Batch filtering is discussed in Section 5.

To prepare for filtering, a dictionary was defined by processing some 1.2 GB of texts consisting of FT91, AP3, all of Foreign Broadcasting FBIS and WSJ-2 collections. These were chosen to be close to the time period 1992-94 as well as content. The dictionary size after stopword removal and Zipf thresholding is about 240K. The filtering collection FT92-94, with long documents segmented into sub-documents, were then processed against this dictionary with no manual classification codes used, only the text portion of each document. The setup was employed to debug codes for mapping physical document order on CDROM to given date order, but not used for training. Training was done using the TREC-7 AP filtering collections, and parameters transferred to this TREC-8 task. We corrected some bugs in our TREC-7 program and also modified our approach.

Many considerations are needed for adaptive filtering. These include defining an initial profile together with an initial selection threshold to start the process, adaptively train the profile to tailor to the type of documents seen so far, dynamically adapt the threshold to select or not select a document for examination, determine how often these changes are to be made, and at the same time attempt to maximize a target utility value. Both adaptation of the filtering profile and that of the threshold are useful. Improved profiles help to separate relevant documents from irrelevant ones better, based on probability or RSV values assigned. Threshold adjustments help to achieve a utility target for the selected documents. Our approach emphasizes on threshold adaptation. Threshold is adjusted periodically after a number of documents have gone through the process and when profiles are updated. Profiles are changed only when a new relevant document has been selected. Moreover, no query expansion was done.

Initial profiles are defined using the raw topic descriptions and our dictionary and term statistics. For document selection when no relevant documents are known, two RSV thresholds Thi and Tlo are defined initially. Documents with $RSV > Thi$ should have high probability of being relevant to a profile, and the opposite is true for documents with $RSV \leq Tlo$. These were set by calculating a profile self-retrieval RSV

(SRSV) [KwGL95]. Each profile is regarded as relevant to its own description when it is considered as a document, and this SRSV value is large. In reality, documents may only overlap partially with a profile and still be relevant, and their RSV's are much less than SRSV. Our two thresholds are defined as: $T_{hi}=hi \cdot SRSV$, $T_{lo}=lo \cdot T_{hi}$, where lo is fixed as 0.8, and hi depends on the utility target F . Typical hi values we used are 0.35 for $F1$ and 0.3 for $F2$ utilities. These values are based on experimentation with TREC-7 filtering discussed later. As filtering proceeds, T_{hi} may be updated, but it is not allowed to fall below T_{lo} if no relevant documents have been selected.

Once the process starts, statistics of term usage is kept for all documents filtered. For documents selected, whether relevant or not, they are stored as a retrieval collection for training purposes. In addition, a running total of the number of documents N that passes through the system, the number examined N_e , and the number found relevant N_r are also kept. This allows us to evaluate an overall average precision $preg=N_r/N_e$ for the user and the proportion of documents examined N_e/N at any time. $preg$ is a global precision indicator. In addition, a local N_r/N_e precision $prel$ for the last two update rounds is also calculated for fine-tuning the adaptation of the threshold.

The update schedule is set to once every $no=2,000$ documents filtered based on experiments with the AP collection. We try to dynamically adjust the RSV threshold T (to determine select or not select a document) based on N , N_r , N_e . Specifically:

```

if (no relevants seen yet)
     $T = T \cdot (1-e)$  when  $T > T_{lo}$  &  $N_e/N < SRT$ 
else {if (change in  $N_r$ ) {
    update profile weights
    recalculate  $T$  using selected docs }
    if (change in  $N_e$ ) {
        if (both  $preg$  &  $prel < G$ )  $T = T \cdot (1+2 \cdot e)$ 
        if (both  $preg$  &  $prel > G$ )  $T = T \cdot (1-e)$  }
    }
```

SRT (selection rate threshold) is set to 0.001 to prevent relaxing T too much if there are too many documents selected already and none is relevant, $e=0.05$ is an adjustment rate. With other parameters fixed, we

hi\G	.4	.45	.5	.55	.6
.3	-271	838	1342	1603	1661
.35	537	1040	1460	1844	1672
.4	431	835	1272	1268	1236

Table 5.1a: Training from AP collection - utility values as a function of hi & G : target $F1=0.4$

hi\G	.3	.35	.4	.45	.5
.27	2070	3028	3575	3590	3304
.3	2967	3823	3772	3893	3494
.35	2848	3314	3564	3346	2821

Table 5.1b: Training from AP collection - utility values as a function of hi & G : target $F2=0.25$

consider the utility performance as a function of G and hi . These are set to achieve maximal utility values according to training parameters from the AP collection as shown in Tables 5.1a,b. We submitted two runs for $F1$: $pir9LF1$ ($hi=.35$, $G=.55$) and $pir9LF1a$ ($hi=.35$, $G=.6$), and two runs for $F2$: $pir9LF2$ ($hi=.3$, $G=.4$) and $pir9LF2a$ ($hi=.3$, $G=.45$).

Results & Discussion

Table 5.2a,b summarize results of the adaptive filtering runs which are named $pir9LF1$ and $pir9LF1a$ respectively for the utility $F1$. $F1$ aims at selecting all documents with a probability of relevance > 0.4 . In addition to $F1$ scores, we tabulate also docs (number of documents selected), $\#rel$ (number of relevant documents selected), precision and recall, and $N+, o, -$ (number of queries that have positive, zero and negative utility). The two runs differ very little.

Comparison with Median			LF1								
FT	>	=	<	score	docs	#rel	Prec	Rec	N+	No	N-
92	31,13	3	16,1	-575 (+278)	520	.93 (576) best submitted	.179	.161	9	7	34
93	14,9	15	21	-438 (+260)	334	46 (629) best submitted	.138	.073	8	14	28
94	27,15	6	17	-254 (+260)	257	52 (647) best submitted	.202	.080	15	14	21
92-4	27,8	2	21	-1268 (+494)	1111	191 (1852) best submitted	.172	.103	12	6	32

Table 5.2a: LF1 Adaptive Filtering for $pir9LF1$

Comparison with Median			LF1								
FT	>	=	<	score	docs	#rel	Prec	Rec	N+	No	N-
92	32,13	2	16,1	-565 (+278)	505	.89 (576) best submitted	.176	.155	10	7	33
93	16,9	15	19	-429 (+260)	332	47 (629) best submitted	.142	.075	10	14	26
94	25,14	6	19	-261 (+260)	253	49 (647) best submitted	.194	.076	13	13	24
92-4	27,8	4	19	-1255 (+494)	1090	185 (1852) best submitted	.170	.100	13	5	32

Table 5.2b: LF1 Adaptive Filtering for $pir9LF1a$
Comparison

with Median				LF2							
FT	>	=	<	score	docs	#rel	Prec	Recl	N+	No	N-
92	12,5	11	27,8	-600 (+435)	1092	123 (576)	.113 best submitted	.214	13	4	33
93	11,5	15	24,7	-174 (+349)	438	66 (629)	.151 best submitted	.105	16	9	25
94	16,8	17	17,5	-39 (+383)	251	53 (647)	.211 best submitted	.082	16	10	24
92-4	13,3	6	31,10	-813 (+990)	1781	242 (1852)	.136 best submitted	.131	20	2	28

Table 5.2c: LF2 Adaptive Filtering for pir9LF2

Comparison with Median				LF2							
FT	>	=	<	score	docs	#rel	Prec	Recl	N+	No	N-
92	12,6	13	25,8	-583 (+435)	1155	143 (576)	.124 best submitted	.248	12	4	34
93	9,5	13	28,9	-181 (+349)	473	73 (629)	.154 best submitted	.116	14	11	25
94	15,8	15	20,6	-54 (+383)	294	60 (647)	.204 best submitted	.093	13	12	25
92-4	10,3	6	34,12	-818 (+990)	1922	276 (1852)	.144 best submitted	.149	17	3	30

Table 5.2d: LF2 Adaptive Filtering for pir9LF2a

This task was not successful as the F1 scores are negative for all years. The learning process for the profile weighting and threshold setting however seem correctly done as the scores get better in successive years. When compared with results from all participants, we have at least 29 instances better or equal to the median out of 50 for all years.

Tables 5.2c,d summarize our filtering runs for the LF2 utility target of 0.25 precision. As previously, utility scores improve year to year, but they are all negative, and results are below median. Filtering the FT collections appears quite a difficult task. Its characteristics seem very different from the AP collection; bringing parameters based on that collection seems not useful. Even the more restrictive parameters set for LF1 do not return positive scores for the LF2 target. However, after results were known, more restrictive parameters were set and we were able to achieve positive utilities of around 65 for F1 and 170 for F2.

6 Batch Filtering and Routing Retrieval

6.1 Pircs and genetic algorithms

The TREC8 filtering and routing tasks were used as a testbed for our research in applying genetic algorithms learning [Gold89,Holl75] in Information Retrieval, in conjunction with the Probabilistic Information Retrieval

Component System (PIRCS). PIRCS itself is a combination of two networks, implementing different retrieval modes, query-focused retrieval (type 1) and document-focused (type 2) retrieval. The user is allowed control over the combination coefficients to fine tune retrieval effectiveness. If these coefficients are set to (0,1) and (1,0), the resulting retrievals will be virtually independent. There are other ways of getting differing retrievals, one of the most effective is varying the term expansion levels.

Given a retrieval system r , which assigns a Retrieval Status Value (RSV_r) to retrieved documents, the output of different retrieval systems can be combined by some function $f(RSV_r)$. A GA can search this space to yield a combination, which is superior to any individual retrieval. A simple function of this type, which we use in the current experiments is linear addition, $\text{sum_of}(a_r * RSV_r)$, where the a_r are arbitrary coefficients. A retrieval of this type, which uses RSV as features instead of term weights, we call second level retrieval.

6.2 Goals for Batch Filtering

Two requirements must be met in order for a batch filtering system to perform well. It must be able to create a profile, which will generate a satisfactory retrieval. In the past TREC meetings there was a high correlation between the best retrievals and the best filtering scores. An additional challenge is to set the retrieval threshold to satisfy the target functions.

6.3 Methodology for Batch Filtering

Fig-1 describes a pictorial representation of the batch filtering procedure.

The FT92 Collection was indexed and statistics were collected by our standard PIRCS system. The collection was divided into two equal parts, a test collection and a training collection. The creation of the final filtered documents was a four-step process.

Step 1) Six retrieval profiles were created from the training subcollection using the Pircs system. They are listed below a run name abbreviation followed by a short description:

- (not1) pircs no training type 1
- (not2) pircs no training type 2
- (pircsb1) pircs type 1 expansion 250
- (pircsb2) pircs type 2 expansion 60
- (pircsf1) pircs type 1 expansion 40
- (pircsf2) pircs type 2 expansion 10

Step 2) Using the six profiles perform retrievals on the FT92 test subcollection. Combination coefficients are computed via a genetic algorithm based learning program. The GA attempts to maximize the average uninterpolated precision.

Step 3) The six profiles are recreated, using the full FT92 Collection. Of course the profiles not1 and not2 are unchanged since they do not learn from relevant documents. The profiles retrieve documents from FT92 and combined using the coefficients from step1. A logistic regression translates the retrieval status values into a probability.

Step 4) The six profiles are now applied to the FT93-94 collection. They are combined using the combination coefficients and transformed by the logistic regression coefficients. The values above the threshold are selected for filtering.

6.4 Selection of Filtering Threshold

There are two reasonable ways to select the cutoff point. One method is to calculate retrieval status value for which the F_i measure yields the maximum. If this occurs at multiple values select one of them. The other is to use logistic regression to transform the retrieval status value into a probability and use the probability for the cutoff. We used the first method prior to TREC7 (and in adaptive filtering) the second since then. Translating the retrieval status value into a probability is also very useful for the user of the system.

Only 43 topics had relevant training documents and we did not submit documents for the other 7. The quality of the training document may not be very good since they were selected by ad-hoc systems. The routing and filtering systems make use of the available judged documents to perform automatic term expansion and training, and consequently uncover more relevant documents. At the TREC7 conference 3301 relevant documents were found for the AP89 collection, while before TREC7 only 1598 were known. Thus the density of relevant documents was over twice as much as was assumed previously. Looking back at TREC7 we observed, that for our filtering run a .25 threshold we would do better at the average of .159 probability and a median of .07, and for the .40 threshold with an average of .298 and a median of .22! Consequently we decided to set the threshold at .30 probability for F1 and .15 for F2. Documents were selected for 30 topics for F1 and for 33 for F2.

The run names for batch filtering documents submitted are pirc9BF1 and pirc9BF2.

6.5 Batch Filtering Results

Subsequently we discovered that our submitted result contained some FT92 documents caused by an incorrect retrieval file. After deleting the FT92 documents from the filtered files, we recomputed the revised scores. Table 6.1 shows the official and revised results.

run	>	=	<
Pirc9BF1 official	19(7)	23(14)	8
Pirc9BF1 revised	25(10)	20(16)	5
Pirc9BF2 official	18(18)	29(15)	3
Pirc9BF2 revised	27(22)	21(16)	2

Table 6.1 Comparison of batch filtering results with median. Number in parenthesis is number of best values.

Compare levels	> , = , <	overall
F1		
.30:.40	12,12,8	+4
.30:.35	8,17,7	+1
.30:.25	12,10,10	+2
.30:.20	12,8,12	0
.30:.15	19,4,9	+10
F2		
.15:.25	17,6,9	+8
.15:.20	14,10,8	+6
.15:.12	13,10,9	+4

Table 6.2. Compare threshold levels for batch filtering.

Threshold	Score
BF1	
.30 official	295
.40	395
.35	395
.30 revised	399
.25	377
.20	415
.15	364
BF2	
.15 official	875
.25	746
.20	856
.15 revised	940
.12	964

Table 6.3. Score at threshold level

We also compared the performance of our revised runs to other threshold levels. Table 6.2 is a query by query comparison and Table 6.3 shows the total score for various levels. It is apparent from the tables that the decision to use lower levels was justified.

7. Routing Track

The focus of our routing runs is to experiment with our genetic algorithm combination of retrievals. The first run pirc9R1 combines 6 retrievals and the second pircs9R2 combines 8. The first submitted routing retrieval pirc9R1 was prepared the same way as the filtering retrieval. We created six profiles using the same expansion and training parameters as described earlier for filtering. They were combined using a GA attempting to maximize the average uninterpolated precision just as for filtering. We also used the same term statistics computed from the FT92 collection. The difference is, that all the relevant documents from FT91 FT92 LA and FBIS were used for training.

For the pirc9R2, two more retrievals were added to the above six to generate the second submitted run. A pure ga based retrieval and a retrieval using backpropagation. For each topic 120 term were selected using our standard pircs system. To these we added 15 positive and 6 negative pairs. The ga optimizes the maximum likelihood measure, thus performing a logistic regression. The backpropagation neural network is a modified version of NevProp a publicly available c program maintained by Phil Goodman of the University of Nevada. No hidden nodes were used for the backpropagation training. In the past we did not have good results with these methods, but the diversity the produce usually enhances the combination.

Routing Retrieval Results

Run name	>	=	<
Pirc9R1	32(8)	8(2)	8
Pirc9R2	22(5)	8(2)	18

Table 7.1 Comparison of routing results with median. Number in parenthesis is number of best values.

The combination Pirc9R1 performed well. The ga and np retrieval did not, and adding it to Pircs9R1 depressed the result. We plan to investigate the cause of this. One possibility is that all evaluated documents were used for training, but the terms added to the query were only based on the relevant documents. These terms may have been underrepresented in the evaluated nonrelevant documents and thus their weight was inflated.

Table 7.2 compares the individual components to the combined retrievals. Max is the hypothetical retrieval that could be achieved if the best retrieval for each query

method	avg	% chg from not2	% chg from Pircsb1
not1	0.2182	23.1%	-47.3%
not2	0.1773	0.0%	-57.2%
Pircsb1	0.4008	126.1%	-3.2%
Pircsf1	0.4140	133.5%	0.0%
Pircsb2	0.3297	86.0%	-20.4%
Pircsf2	0.3273	84.6%	-20.9%
max	0.4670	163.4%	12.8%
Pirc9R1	0.4316	143.5%	4.3%
Pirc9R2	0.3990	125.0%	-3.6%

Table 7.2 Individual retrieval results.

	Pircsb1	pirc9R1	% chng
Rel_ret	1214	1203	-0.91%
at 0.00	0.7207	0.7380	2.40%
at 0.10	0.6463	0.6801	5.23%
at 0.20	0.5923	0.6245	5.44%
at 0.30	0.5410	0.5737	6.04%
at 0.40	0.5008	0.5132	2.48%
at 0.50	0.4425	0.4539	2.58%
at 0.60	0.3770	0.3941	4.54%
at 0.70	0.3096	0.3315	7.07%
at 0.80	0.2586	0.2679	3.60%
at 0.90	0.1816	0.1917	5.56%
at 1.00	0.1319	0.1425	8.04%
	0.4140	0.4316	4.25%
Precision:			
At 5 docs:	0.5080	0.4920	-3.15%
At 10 docs:	0.4020	0.428	6.47%
At 15 docs:	0.3533	0.376	6.43%
At 20 docs:	0.3190	0.328	2.82%
At 30 docs:	0.2740	0.2747	0.26%
At 100 docs:	0.1462	0.1516	3.69%
At 200 docs:	0.0940	0.0941	0.11%
At 500 docs:	0.0456	0.045	-1.32%
At 1000 docs:	0.0243	0.0241	-0.82%
Exact:	0.3985	0.4168	4.59%

Table 7.3 Routing Effectiveness Levels

was known. Pirc9R2 improved 4.3% over the best retrieval Pircsf1 but it did not reach the performance of max. Table 5.3 is a more detailed comparison of Pirc9R2 with the best performing individual retrieval Pircsf1.

8 Conclusion

In TREC8 experiments we continued to demonstrate that our PIRCS system consistently return competitive results. For ad-hoc retrieval, multiple techniques such as combination of retrieval lists (data fusion), collection enrichment and 2-stage pseudo-feedback all can cooperatively boost effectiveness to the best level. For query track, we showed the importance of term choices in query formulation. For adaptive filtering track, we showed that minimally storing only selected documents can enable us to do filtering and adaptive threshold setting. Our utility scores are negative possibly due to difficulties in acquiring training data for this FT collection. Batch filtering and routing continues to do well.

References

- Gold89 Goldberg, D. E. "Genetic Algorithms in Search Optimization & Machine Learning" Addison-Wesley 1989.
- HiKr99 Hiemstra, D & Kraaij, W. "Twenty-One at TREC-7: Ad-hoc and Cross-Language Track" In: NIST Special publication 500-242. pp.227-238, 1999.
- Holl75 Holland, J.H. Adaptation in Natural and Artificial Systems; University of Michigan Press: Ann Arbor, MI, 1975.
- KwCh98 Kwok, K.L. & Chan, M. "Improving Two-Stage Ad-Hoc Retrieval for Short Queries". In: Proc. 21st SIGIR'98Conference. pp.250-256, 1998.
- KwGL95 Kwok, K.L, Grunfeld, L & Lewis, D.D. "TREC-3 Ad-hoc, Routing Retrieval and Thresholding Experiments using PIRCS" In: NIST Special Publication 500-225. pp.247-255.
- Kwok95 Kwok, K.L. " A Network Approach to Probabilistic Information Retrieval". Journal ACM Transactions on Information Systems. Volume 13, pp.324-353 1995
- VoHa99 Voorhees, E.M & Harman, D.K. Overview of the Seventh Text REtrieval Conference (TREC-7). In: NIST Special Publication 500-242. pp.1-23; 1999.

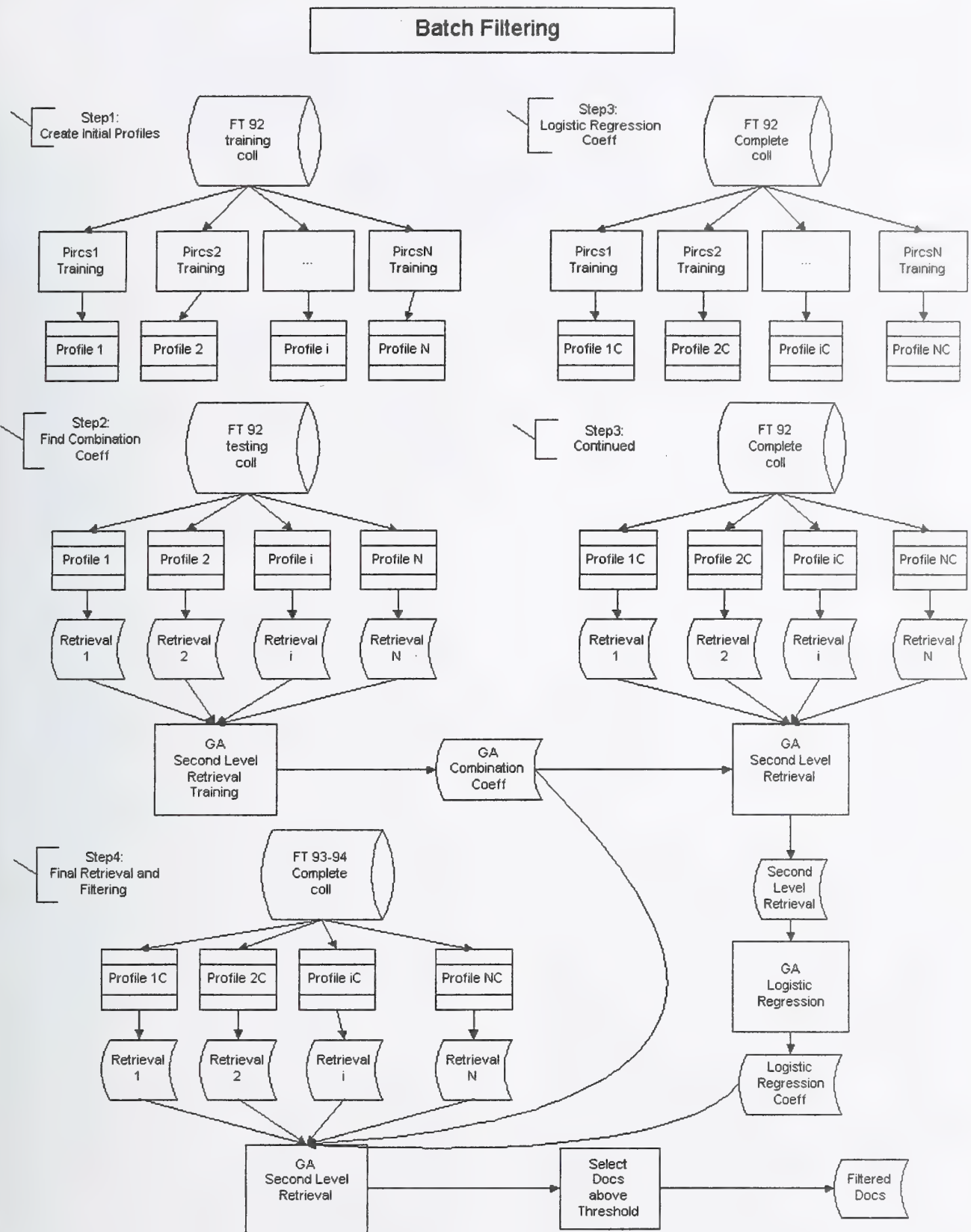


Fig.1: Batch Filtering System Flowchart

Report on the TREC-8 Experiment: Searching on the Web and in Distributed Collections

Jacques Savoy, Justin Picard

Institut interfacultaire d'informatique
Université de Neuchâtel (Switzerland)

e-mail: {Jacques.Savoy, Justin.Picard}@seco.unine.ch

Web page: <http://www-seco.unine.ch/info>

Summary

The Internet paradigm permits information searches to be made across wide-area networks where information is contained in web pages and/or whole document collections such as digital libraries. These new distributed information environments reveal new and challenging problems for the IR community. Consequently, in this TREC experiment we investigated two questions related to information searches on the web or in digital libraries: (1) an analysis of the impact of hyperlinks in improving retrieval performance, and (2) a study of techniques useful in selecting more appropriate text databases (database selection problem encountered when faced with multiple collections), including an evaluation of certain merging strategies effective in producing, single, ranked lists to be presented to the user (database merging problem).

Introduction

There is an increasing interest in hypertext systems, digital libraries and in effective web searching [Bernes-Lee 94]. Due to the huge number of pages and links, browsing cannot be viewed as an adequate searching process, even with the introduction of tables of contents or other classified lists (e.g., Yahoo!) [Alschuler 89]. As a result, effective query-based mechanisms for accessing information will always be needed [Halasz 88]. The search engines currently available on the web [Leighton 99], [Gordon 99] are hardly able to cover all available information [Lawrence 99] and they are characterized by many drawbacks [Hawking 99a]. Moreover, in order to enhance their retrieval effectiveness, most of them ignore hypertext links. Recent works in IR on the web seem to acknowledge that hyperlink structures can be very valuable in locating information [Marchiori 97], [Kleinberg 98], [Brin 98],

[Chakrabarti 99], [Bharat 98]; and according to Chakrabarti et al. [99]:

"Citations signify deliberate judgment by the page author. Although some fraction of citations are noisy, most citations are to semantically related material. Thus the relevance of a page is a reasonable indicator of the relevance of its neighbors, although the reliability of this rule falls off rapidly with increasing radius on average. Secondly, multiple citations from a single document are likely to cite semantically related documents as well." [Chakrabarti 99, p. 550-551]

With small variations, similar hypotheses are also cited by other authors [Kleinberg 98], [Bharat 98]. Our previous studies on citation schemes [Savoy 94], [Savoy 96a], [Savoy 97], [Picard 98] tend to suggest however that citation information might improve average precision, but only on the order of 5% to 8% when used with good retrieval schemes.

The first chapter of this paper verifies whether or not hyperlinks improve retrieval effectiveness. In the second chapter, we describe experiments on the ad hoc track. In this case, we acknowledge that currently it is becoming more and more difficult to store and manage the growing document collections within a single computer. Recent advances in network technology do however allow us to disseminate information sources by partitioning a single huge corpus (or distributing heterogeneous collections) over a local-area network (Intranet). Most retrieval mechanisms currently proposed however are based on conventional IR models [Salton 89], and where a centralized document collection is assumed.

To access these distributed collections, our IR model sends a request to several separate and selected text databases (each having its own search

engine) on the one hand, and on the other, it implies merging of the resultant output lists in the form of an "optimal" single list to be presented to the user. Thus, our approach must address the following problems [Dreilinger 97]:

- selecting the appropriate set of information servers to which the query will be sent (collection selection problem);
- converting the information need into a format readable by the selected search engines (e.g., based on the Z90.50 protocol for inter-system retrieval [Kahle 93], or on STARTS model [Gravano 97]);
- selecting and sorting the result lists obtained by different information servers to form a unique result list (database merging problem).

Chapter two of this paper reflects our interest in addressing the first and last problems, both of which may be viewed as serious. To evaluate our hypothesis, we used the SMART system as a test bed for implementing the various vector-processing weighting schemes along with the OKAPI probabilistic model [Robertson 95]. This year our experiments were conducted on an Intel Pentium III/450 (cache: 1MB, memory: 256 MB, disk: RAID0 EIDE with 2 x 27 GB) and all of our experiments are fully automated.

1. Small Web Track

Our participation with the web track addresses the following question: do hyperlinks provide useful evidence in enhancing a search engine's retrieval?

Some statistics describing the web collection are listed in Table 1 and other characteristics are described in [Hawking 99b]. Of note is that this corpus possesses 1,171,795 hyperlinks leading to an average of 4.73 hyperlinks per page (used primarily for navigational purpose across the web site). Relative to the Web which is currently estimated to contain about 800 million web pages [Lawrence 99], our test collection might be viewed as being relatively small. There is consequently the risk that a large portion of the hyperlinks between pages having different URLs (defined as the IP number) will be unusable, because the destination node will very likely be outside of the collection. According to our computations, there were 2,797 hyperlinks to pages on different hosts,

representing 0.24% of the total. Moreover, most of these links were grouped in clusters (e.g., one or a few web pages from one site point to one or a few web pages from another site).

In order to proceed with our evaluation, we used the non-interpolated average precision at eleven recall values, based on 1,000 retrieved items per request. To determine whether or not a given search strategy is better than another, we need a decision rule. The following rule of thumb could provide serve as such a rule: a difference of at least 5% in average precision is generally considered significant and a 10% difference is considered material [Sparck Jones 77, p. A25].

From the original WWW pages, we retained the following logical sections: <title>, <h1>, <center>, <big> and for delimiting document boundaries: <docno>. Thus, the most common tags <P> (or <p>, together with </P>, </p>) have been removed. Text delimited by the tags <DOCHDR>, </DOCHDR> were also removed. A classical stemming procedure was applied and stopwords were removed.

1.1. Pseudo-Relevance Feedback

It is recognized that pseudo-relevance feedback (blind expansion) is a useful technique for enhancing retrieval effectiveness. Thus, we have evaluated the OKAPI search model with and without query expansion to verify whether or not this technique might improve retrieval performance when faced with different query formulations (such technique is known to be time-consuming). In this study, we have adopted Rocchio's approach [Buckley 96] with $\alpha = 0.75$, $\beta = 0.75$ and the system is allowed to add 17 search terms to the original query during feedback which are extracted from the 30-best ranked documents. The resulting retrieval effectiveness is depicted in Table 2a.

Pseudo-relevance feedback results in satisfactory and significant enhancement over baseline performance. This improvement is more important when dealing with short queries (2.4 search terms in average). However Table 2b shows that retrieval time is significantly increased with procedure.

Size (in MB)	2,000 MB
# of web pages extracted from 969 URLs	247,491
# of distinct indexing terms in the collection	1,850,979
# of distinct index terms / web page	
mean	218.25
standard error	326.42
median	125
maximum	22722
minimum	1
# of indexing terms / web page	
mean	554.295
standard error	1402.86
median	213
maximum	179,303
minimum	1
time required to build the inverted file	
(user time)	26:28
elapsed time	1:44:44
# of relevant web pages (100 queries)	8,868
from Topic #351 to Topic #400	6,589
from Topic #401 to Topic #450	2,279

Table 1: Small Web Collection Statistics

Model \ Query	Precision (% change)		
	Title	Title & Desc	Title, Desc & Narr
doc = OKAPI, query = NPN	23.49	27.39	30.34
with query expansion	29.55 (+25.80%)	31.36 (+14.49%)	30.74 (+1.32%)

Table 2a: Average Precision of Blind Query Expansion

	Search Time in sec. (% change)		
search time (original) / request	0.3033	0.5279	0.8185
search time (expand) / request	4.570 (+1406%)	4.748 (+999%)	5.138 (+527%)

Table 2b: Search Time per Request (in sec.)

1.2. Hyperlinks

Based on our previous studies on citation schemes [Savoy 96a], [Savoy 97], [Picard 98], we have taken hyperlinks into account to hopefully improve retrieval effectiveness. The common point of spreading activation techniques [Cohen 87] used in our previous works [Savoy 96a], [Savoy 97] and the probabilistic argumentation systems (PAS) [Picard 98] used here is to consider links as a way of improving the initial ranking of documents.

Instead of directly trying to use the hyperlinks for retrieval, we believe it is better to understand how they relate to the relevance of a document, and to estimate to what degree this relationship holds (Section 1.2.1). Then we will apply the spreading activation technique and PAS to integrate these links into the retrieval process (Section 1.2.2). Finally we will draw some conclusions on the potentiality offered by links for retrieval on the web, in regard of the experimental results obtained (Section 1.2.3).

1.2.1. Relationships between Hyperlinks and Relevance

The hypothesis underlying our experiments is that hyperlinks contain some information about relevance. Before starting experiments, it is therefore certainly advisable to have a better understanding of how and to what degree links are sources of evidence about relevance. This can be enlightening and can help in determining which techniques better fit the particular situation at hand.

Our main idea in using hyperlinks is to consider that they may propagate some score or probability. But when should a link propagate information to other documents? Clearly if the document is not relevant, this will not tell us much about the linked documents. However if it is relevant, one should expect that there is some probability that the linked documents will also be relevant, or in other words, that the link is "valid". Obviously, the higher this probability, the greater the link's information about relevance. It would then be interesting to estimate this probability using a training set, in order to get an idea on what can (and cannot) be expected from links. For this purpose we used Topics #351 to #400.

A possible technique for estimating this link probability is the following. For each relevant document, we compute the fraction of linked documents that are themselves relevant, then we compute the average of this fraction on all queries (Algorithm 1). An objection to this method is that some documents are linked to more than one relevant document, and will have a higher probability of being relevant. To avoid an upward biased estimate, we exclude these documents from computation, and compute the probability in the same way as Algorithm 1 (Algorithm 2). Finally, the link probability might vary largely between queries, mostly because the number of relevant documents can vary by one or even two orders of magnitude. In order to keep a few queries from dominating the computation, we take Algorithm 2 but compute the

median instead of the mean (Algorithm 3). The resulting probability estimates are given in Table 3.

From data depicted in Table 3, one can find that depending on the algorithm used, the estimate may vary greatly. The experiments presented in the next subsection make direct use of this probability, and work better for the smallest estimates found with Algorithm 3. This finding strongly suggests that this value is a better estimate of the link probability. It is lower than equivalent estimates found with the CACM collection (based on bibliographic references rather than hyperlinks).

Other experiments, which are not displayed here evaluated the impact on a document's probability of relevance, given that it is linked or not to one of the five best ranked documents. It seems that being linked to one of the five best ranked documents does not affect the probability of relevance for the 25 best-ranked documents, and increases it slightly for higher ranks. This result tends to confirm that hyperlinks should have a small impact on retrieval effectiveness.

1.2.2. Experiments

For the two techniques, we only considered only links from/to the 50 best-ranked documents. We took the initial rank and score of each document, and computed a retrieval status value (spreading activation) or a degree of support (PAS), after the integration of link information. Documents were then reranked according to this new score/degree of support.

We first experimented with the simple technique of spreading activation. In that method, the degree of match of a document and a query, as initially computed by the IR system (denoted $s(D)$), is propagated to the linked documents through a certain number of cycles using a propagation factor. We used a simplified version with only one cycle and a fixed propagation factor λ for all links of a certain type (incoming or outgoing). In that case, the final retrieval status value (RSV) of a document D linked to n documents is:

Estimation Method	Incoming Links	Outgoing Links
Algorithm 1	0.145	0.106
Algorithm 2	0.066	0.090
Algorithm 3	0.062	0.051

Table 3: Probability Estimates of Links for Our Three Algorithms

$$RSV(D) = s(D) + \lambda \cdot \sum_{i=1}^n s(D_i)$$

We experimented with several values of the propagation factor λ . Even for the smallest values of λ , a deterioration in retrieval effectiveness resulted, and this deterioration increased monotonically for increasing parameter values. This tends to show that simple and intuitive techniques, which produced satisfactory results in other retrieval environments, do not seem to perform well in this situation. It is our opinion that hyperlinks seem to provide less information than do the bibliographic references or co-citation schemes used in our previous studies.

In a second set of experiments, we used probabilistic argumentation systems (PAS) [Picard 98]. For this study, we used a simplified version of our approach where a document's degree of support (and thus its rank) can be affected only by its direct neighbors. In that case we do not need to keep track of inferences, and can derive a simple formula which can be understood as a more refined way of spreading activation. Instead of propagating a document's score, we propagated its probability of being relevant. This probability was multiplied by the probability of the link, denoted $p(\text{link})$, and then assessed according to Section 1.7.1. To compute the probability of relevance of a document given its rank $p(D|\text{rank})$, we fitted a logistic regression [Bookstein 92] to its rank for the set of training Topics #351 to #400.

The individual contribution of a document D_i is then $p(D_i|\text{rank}) \cdot p(\text{link})$, instead of $s(D_i) \cdot \lambda$ used with the spreading activation technique. In the case where a document had more than one source of evidence indicating relevance, the spreading activation technique summed the individual contributions. In the PAS technique, the initial rank of a document and the contribution of each linked document were considered as different sources of

evidence. A source of evidence e_i has a certain probability $p(e_i)$ to being valid, and the degree of support (DSP) of a document is computed as the probability that at least one of the source of evidence is valid.

$$dsp(D) = 1 - \prod_{i=1}^n (1 - (p(e_i)))$$

Experiments using all incoming or outgoing links did not demonstrate any improvement. We then decided to include only the most important sources of evidence: the initial rank of the document D , the best incoming document D_{in} and the best outgoing document D_{out} .

$$\begin{aligned} dsp(D) = & 1 - (1 - (p(D|\text{rank}))) \cdot \\ & (1 - p(D_{in}|\text{rank}) \cdot p(\text{link}_{in})) \cdot \\ & (1 - p(D_{out}|\text{rank}) \cdot p(\text{link}_{out})) \end{aligned}$$

For the values of $p(\text{link}_{in})$ and $p(\text{link}_{out})$ computed with Algorithm 3, we obtained improvements of between 1% to 1.5% for Topics #351 to #400. Other values of these probabilities did not yield higher retrieval effectiveness. The results obtained on Topics #401 to #450 are given in Table 4. However, hyperlinks may be valuable for other purposes; for example, citation information have been used to define co-citation clusters for better visualizing the relationships between disciplines, fields, specialties, and individuals papers [Small 99].

1.2.3. Official Web Runs

Our official run (UniNEW2Ct, content-only) resulted in an average precision of 31.50, 41 times above the median and for the two queries (#424, #434), it displays the best results. The related official run (UniNEW2Link, content & links) shows a small but not significant degradation in average performance.

Official Run Name	Average Precision	# \geq Median	# Best
UniNEWCt	27.39	34	0
UniNEWLink	27.47 (+0.29%)	44	3
UniNEW2Ct	31.50	41	2
UniNEW2Link	31.37 (-0.41%)	44	9

Table 4: Summary of Our Official Runs for the Web Track

1.3. Summary of Results

The various experiments carried out within the web track showed that:

- Hyperlinks do not result in any significant improvement (at least as implemented in this study). Link information seems to be marginally useful for retrieval on the web, especially when the retrieval system produces relatively high retrieval effectiveness;
- Pseudo-relevant feedback techniques (blind query expansions) result in significant improvement but they increase search times (by a factor of ten in our implementation);

2. Distributed Collections

To evaluate the retrieval effectiveness of our distributed IR model, we formed four separate sub-collections according to the source of the available documents. Table 5 summarizes various statistics about these four sub-collections and depicts general statistics of the collection named TREC8.

In this study with our distributed IR model, we assumed that each search engine used the same indexing scheme and the same retrieval procedure. Such a distributed context reflects a local area network more closely than does the Internet where different search engines may collaborate to search for information [Le Calvé 99]. Our approach may be more closely identified by the following characteristics. In the first stage and based on the current query, our IR model must select the more appropriate set of sub-collections on which the search will be done (Section 2.2, see also [Callan 95], [Xu 98], [Fuhr 99], [Hawking 99a]). Based on this selection procedure, the query will be sent to the

selected text databases and depending on the results, the system will merge them into a single result list to be presented to the user (Section 2.3).

Before describing the collection selection and the collection fusion approaches, Section 2.1 identifies retrieval effectiveness measures achieved by various search models with the whole collection and with each of our four sub-collections. These results from this evaluation are useful in our context, since our investigations are not limited to a single search model. Rather, they may be used with different search strategies, leading hopefully to a more general conclusion.

2.1. Environment

In order to obtain a rough idea regarding the retrieval effectiveness of our sub-collections compared to that of the whole TREC8 collection, we conducted different experiments using various weighting schemes, the vector-processing model (denoted using SMART parlance, see Appendix 1) and the OKAPI probabilistic model. To adjust the underlying parameters of the OKAPI search model, we used $adv1 = 750$, $b = 0.9$, $k1 = 2$. For the LNU weighting scheme, we set the parameters to: $slope = 0.2$ and $pivot = 150$.

The results depicted in Table 6 show that the retrieval effectiveness of each sub-collections was higher than that of the whole collection, but it must be remembered that the number of queries and the number of relevant documents were not the same across all sub-collections. We do think however that this information indicates that a good selection procedure may enhance the retrieval effectiveness compared to the average precision achieved from using the whole collection.

Collection	FT	FR	FBIS	LA Times	TREC8
Size (in MB)	564 MB	395 MB	470 MB	475 MB	1,904 MB
# of documents	210,158	55,630	130,471	131,896	528,155
# of distinct index terms / document					
mean	124.4	131.16	141.56	158.46	136.84
standard error	93.26	127.95	125.04	124.11	114.54
median	101	128	107	122	108
maximum	3,050	23,517	5,677	5,040	23,515
minimum	6	2	6	10	2
# of indexing terms / document					
mean	195.62	320.11	267.2	262.86	240.89
standard error	172.66	1,128.3	598.82	248.5	501.35
median	151	289	168	184	171
maximum	13,761	211,944	61,300	16,100	211,934
minimum	6	2	10	10	2
# of distinct indexing terms	375,499	196,220	502,099	337,492	1,008,463
min idf	$0.092 \cdot 10^{-4}$	$0.1845 \cdot 10^{-4}$	$0.0805 \cdot 10^{-4}$	$0.3794 \cdot 10^{-4}$	$6905.49 \cdot 10^{-4}$
max df	210,156	55,629	130,470	131,891	264,765
time to build the inverted file	32:01	12:55	24:33	33:06	
from Topics #301 to #450					
# of relevant documents	4,903	844	4,410	3,535	13,692
# of queries	144	69	116	143	150
from Topics #301 to #400					
# of relevant documents	3,233	638	2,743	2,350	8,964
# of queries	95	50	60	98	100
from Topics #401 to #450					
# of relevant documents	1,670	206	1,667	1,185	4,728
# of queries	49	19	43	45	50

Table 5: Statistics on TREC8 Collections

Collection	Precision				
	FT	FR	FBIS	LA TIMES	TREC8
	49 queries 1,670 rel.	19 queries 206 rel.	43 queries 1,667 rel.	45 queries 1,185 rel.	50 queries 4,728 rel.
Model					
OKAPI - NPN	40.00	38.27	33.75	31.11	29.65
LNU - LTC	34.17	25.64	25.50	26.94	24.57
ATN - NTC	33.96	35.56	30.65	27.90	26.25
NTC - NTC	18.63	17.35	13.92	15.79	13.09
LTC - LTC	23.60	30.85	22.59	21.15	17.49
LNC - LTC	25.28	23.75	20.32	24.67	19.40
LNC - LNC	18.26	11.24	12.42	21.86	12.05
ANC - LTC	24.39	26.20	20.53	23.04	17.51
NNN - NNN	6.97	3.25	2.71	6.90	1.61
BNN - BNN	9.00	5.74	5.04	3.65	3.12

Table 6: Average Precision of Isolated Collections (Query = Title, Desc & Narr)

2.2. Selection Strategy

As a first attempt to define a selection procedure, we wanted a strategy that, based on the current request, might produce a binary outcome, specifying whether or not the underlying sub-collection contained pertinent document(s) or not. Our challenge was to define an automatic procedure that would answer to the question "Does this collection (with its search engine) provide a satisfactory answer (at least one relevant document) to this question?". Therefore, the expected answer was not an integer value specifying the number of records to be retrieved from the underlying sub-collection but a binary outcome. With such a procedure, the computer could be aware of the limits of its knowledge, knowing when it does not know.

In this study, we wanted to verify whether or not past requests might be useful sources of evidence for such selection purposes. To achieve this, we defined a selection procedure based on the k-nearest neighbors (k-NN) technique that works as follows (see also [Voorhees 95], [Voorhees 96], [Savoy 96b]).

For each new topic Q , the system found the k nearest neighbors in the set of all existing requests Q_j , $j = 1, 2, \dots, m$ ($m = 149$, $k = 3$, cosine measure). The three-best ranked past requests were retrieved and the system determined whether or not, for those three requests, the underlying sub-collection contained any pertinent records. Based on the majority rule, the system might decide whether or not to conduct a search into the underlying sub-collection.

During the testing stage of our system (based on Topics #301 to #400), we noticed that the FT sub-collection contained pertinent information for 95 queries out of a total of 100, while the LA sub-collection had relevant documents for 98 queries. Therefore, we decided, for each new request (Topics #401 to #450), to search in both the FT and LA sub-collections without considering our selection procedure. On the other hand, based on the training requests (Topics #301 to #400), the FR collection

may produce relevant information for 50 queries and the FBIS sub-collection for 60. Therefore, we apply our selection procedure only for these two sub-collections.

The complete evaluation of our decision rule is given in Tables 7. First, in Table 7a, the decision taken by the system is represented in the rows while the true state of Nature is depicted in the columns. For example, the number "8" indicates that 8 times the system decided to retrieve information from the FR sub-collection and these decisions were correct (true positive). Of course, our selection procedure produces also errors, e.g., for the FR collection, it decided four times to conduct a search while this corpus did not hold any relevant information (false positive).

As an overall correctness indicator, we would compute the accuracy of the decision rule by dividing the number of correct answers (true positive + true negative) by the number of cases. Other evaluation measures are depicted in Table 7b. From these results, it can be seen that the k-nearest neighbors (k-NN) technique does not result in a satisfactory overall performance. Our selection rule is not very sensitive and often fails to conduct a search when it is appropriate.

Our selection procedure is thus far from perfect and the retrieval performance it achieves is also affected by its poor decision-making performance, as shown in the last column of Table 8 (merging according to the raw-score strategy, see Section 2.3). Indicated in the second column of this table is the average precision achieved when all the documents formed a single huge collection (baseline). Depicted in the third column is the average performance we might expect when, for all requests, we decided to search in all the sub-collections and merged the four result lists based on the raw-score merging strategy (see Section 2.3). Under the heading "Optimal Selection" are listed the average precision obtained using an error-free (perfect) selection procedure, ignoring sub-collections having no relevant information for a given query (merging done by the raw-score scheme).

FR	true state		FBIS	true state	
prediction	do retrieve	no retrieve	prediction	do retrieve	no retrieve
do retrieve	8	4	do retrieve	13	2
no retrieve	11	27	no retrieve	30	5
total	19	31	total	43	7

Table 7a: Evaluation of Our Selection Procedure

Measure \ Collection	FR	FBIS
Accuracy (# correct decisions / # cases)	35 / 50 = 0.7	18 / 50 = 0.36
Sensitivity (# true positive / # positive cases)	8 / 19 = 0.42	13 / 43 = 0.302
Specificity (# true negative / # negative cases)	27 / 31 = 0.871	5 / 7 = 0.714

Table 7b: Various Evaluation Measures of Our Selection Rule

Strategy	Precision (% change)			
	Single Collection	No Selection	Optimal Selection	Our Selection Approach
OKAPI-NPN	29.65	27.39 (-7.62%)	29.31 (-1.15%)	22.64 (-23.64%)
LNU - LTC	24.57	23.75 (-3.33%)	24.55 (-0.08%)	19.25 (-21.65%)
ATN - NTC	26.25	24.64 (-6.13%)	26.18 (-0.27%)	20.51 (-21.87%)
NTC - NTC	13.09	12.89 (-1.53%)	13.59 (+3.82%)	11.56 (-11.69%)
LTC - LTC	17.49	16.26 (-7.03%)	17.49 (0.00%)	13.61 (-22.18%)
LNC - LTC	19.40	19.00 (-2.06%)	19.81 (+2.11%)	15.45 (-20.36%)
LNC - LNC	12.05	12.31 (+2.16%)	13.05 (+8.30%)	10.13 (-15.93%)
ANC - LTC	17.51	17.47 (-0.23%)	18.32 (+4.63%)	13.88 (-20.73%)
NNN - NNN	1.61	1.60 (-0.62%)	2.62 (+62.73%)	3.31 (+105.6%)
BNN - BNN	3.12	3.15 (+0.96%)	3.74 (+19.98%)	2.22 (-28.85%)

Table 8: Average Precision of Various Selection Strategies and Merging Done by the Raw-Score Strategy (Query = Title, Desc & Narr)

2.3. Collection Merging

Recent works have suggested that some solutions to the merging of separate answer lists may be obtained from distributed information services. As a first approach, we might assume that each database contains approximately the same number of pertinent items and that the distribution of the relevant documents is the same across the servers' answers. Based only on the ranking of retrieved records, we might interleave the results in a round-robin fashion. According to previous studies [Voorhees 95], [Callan 95], the retrieval effectiveness of such interleaving schemes is around 40% below the performance achieved by a single retrieval scheme working, with a single huge collection representing the entire set of documents.

The third column of Table 9 confirms this finding but to a lesser extent (around -27%).

In order to take account of the score achieved by the retrieved document, we might formulate the hypothesis that each information server applies the same or a very similar search strategy and that the similarity values are therefore directly comparable [Kwok 95], [Moffat 95]. Such a strategy, called raw-score merging, produces a final list, sorted by the retrieval status value computed by each separate search engine. However, as demonstrated by Dumais [94], collection-dependent statistics in document or query weights may vary widely among sub-collections; and therefore, this phenomenon may invalidate the raw-score merging hypothesis. The fourth column of Table 9 indicates the retrieval effectiveness of such merging approach, showing a relatively interesting performance in our case

(degradation of around -2.5%). Thus, the raw-score merging seems to be a simple and valid approach when a huge collection is distributed across a local-area network and operated within the same retrieval scheme.

As a third merging strategy, we may normalize each sub-collection's retrieval status value (RSV) by dividing it by each result list's maximum RSV. The fifth column of Table 9 shows its average precision, representing surprisingly poor retrieval effectiveness (average reduction of -25%).

Finally, we suggest using the logistic regression approach to resolve merging problems that have shown interesting performance levels when merging heterogeneous result lists produced by different search models where only ranks of the retrieved items are available as a key for merging [Le Calvé 99]. In the current case, the explanatory variables are the logarithm of the rank of the retrieved item together with its score. The average precision achieved by this method shown in the last column of Table 9 is similar to the raw-score merging strategy.

2.4. Official Ad Hoc Runs

Our first official run (UniNET8St, ad hoc, automatic, short queries) resulted in an average precision of 29.06, 38 times greater than the median and for two queries (#403, #416), it revealed the best results. Our second official run (UniNET8Lg, ad hoc, automatic, long queries) resulted in an average precision of 31.38, 40 times greater than the

median and for four queries (#416, #429, #431, #438), it revealed the best results. Both results were obtained using the OKAPI retrieval scheme with blind query expansion ($\alpha = 0.75$, $\beta = 0.75$) and the system was allowed to add 50 search terms to the original query during feedback, with added terms extracted from the 5-best ranked documents.

2.5. Conclusion

When dealing with distributed collections across a local area network and using the same retrieval model for all these sub-collections, our experiments show that:

- Selection procedure, based on k-NN technique, does not seem to be worthwhile approach;
- Based on various search strategies, it seems that the raw-score approach might be a valid first attempt for merging result lists provided by the same retrieval model.

Acknowledgments

The authors would like to thank C. Buckley from SabIR for giving us the opportunity to use the SMART system, without which this study could not have been conducted. This research was supported by the SNSF (Swiss National Science Foundation) under grant 20-50'578.97 and 20'55'771.98). The authors would also like to thank Yves Rasolofo for his help computing the logistic regression coefficient estimates.

Strategy Model	Precision (% change)				
	Single Collection	Round -Robin	Raw-Score Merging	Normalized Score	Logistic Regression
OKAPI-NPN	29.65	21.61 (-27.12%)	27.39 (-7.62%)	22.66 (-23.58%)	26.83 (-9.51%)
LNU - LTC	24.57	17.72 (-27.88%)	23.75 (-3.33%)	17.35 (-29.38%)	23.86 (-2.89%)
ATN - NTC	26.25	19.07 (-27.35%)	24.64 (-6.13%)	19.74 (-24.80%)	23.29 (-11.28%)
NTC - NTC	13.09	9.25 (-29.33%)	12.89 (-1.53%)	9.59 (-26.74%)	12.64 (-3.44%)
LTC - LTC	17.49	13.12 (-24.99%)	16.26 (-7.03%)	12.96 (-25.90%)	16.67 (-4.69%)
LNC - LTC	19.40	13.69 (-29.43%)	19.00 (-2.06%)	14.11 (-27.27%)	18.82 (-2.99%)
LNC - LNC	12.05	9.40 (-21.99%)	12.31 (+2.16%)	8.71 (-27.72%)	12.75 (+5.81%)
ANC - LTC	17.51	13.40 (-23.47%)	17.47 (-0.23%)	13.21 (-24.56%)	17.52 (+0.06%)
NNN - NNN	1.61	2.76 (+71.43%)	1.60 (-0.62%)	0.77 (-52.2%)	3.54 (+119.88%)
BNN - BNN	3.12	2.71 (-13.14%)	3.15 (+0.96%)	2.34 (-25.0%)	2.78 (-10.90%)

Table 9: Average Precision of Various Merging Strategies (Query = Title, Desc & Narr)

Official Run Name	Average Precision	# \geq Median	# Best
UniNET8St	29.06	38	2
UniNET8Lg	31.38 (+7.98%)	40	4

Table 10: Summary of our Official Ad Hoc Runs

References

- [Alschuler 89] L. Alschuler : Hand-Crafted Hypertext - Lessons from the ACM Experiment. In E. Barrett (Ed.), *The Society of Text, Hypertext, Hypermedia, and the Social Construction of Information*, (pp. 343-361), The MIT Press, Cambridge (MA), 1989.
- [Bernes-Lee 94] T. Bernes-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, A. Secret: *The World-Wide Web*. Communications of the ACM, 37(8), 1994, 76-82.
- [Bharat 98] K. Bharat, M. Henzinger: Improved Algorithms for Topic Distillation in Hyperlinked Environments. Proceedings of ACM-SIGIR'98, Melbourne (Australia), August 1998, 104-111.
- [Bookstein 92] A. Bookstein, E. O'Neil, M. Dillon, D. Stephens: Applications of Loglinear Models for Informetric Phenomena. *Information Processing & Management*, 28(1), 1992, 75-88.
- [Brin 98] S. Brin, L. Page: The Anatomy of a Large-Scale Hypertextual Web Search Engine. Proceedings of WWW8, Brisbane (Australia), April 1998, 107-117. <http://google.stanford.edu>.
- [Buckley 96] C. Buckley, A. Singhal, M. Mitra, G. Salton: New Retrieval Approaches using SMART. Proceedings of the TREC'4, Gaithersburg (MD), NIST publication 500-236, 1996, 25-48.
- [Callan 95] J. P. Callan, Z. Lu, W. B. Croft: Searching Distributed Collections with Inference Networks. Proceedings of the ACM-SIGIR'95, Seattle (WA), 1995, 21-28.
- [Chakrabarti 99] S. Chakrabarti, M. Van den Berg, B. Dom: Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. Proceedings of WWW8, Toronto (ON), May 1999, 545-562.
- [Cohen 87] P. R. Cohen, R. Kjeldsen: Information Retrieval by Constrained Spreading Activation in Semantic Networks. *Information Processing & Management*, 23(4), 1987, 255-268.
- [Dreilinger 97] D. Dreilinger, A. E. Howe: Experiences with Selecting Search Engines using Metasearch. *ACM Transactions on Information Systems*, 15(3), 1977, 195-222.
- [Dumais 94] S. T. Dumais: Latent Semantic Indexing (LSI) and TREC-2. Proceedings of TREC'2, Gaithersburg (MD), NIST Publication #500-215, 1994, 105-115.
- [Fuhr 99] N. Fuhr: A Decision-Theoretic Approach to Database Selection in Networked IR. *ACM Transactions on Information Systems*, 1999, to appear.
- [Gordon 99] M. Gordon, P. Pathak: Finding Information on the World Wide Web: The Retrieval Effectiveness of Search Engines. *Information Processing & Management*, 35(2), 1999, 141-180.
- [Gravano 97] L. Gravano, K. Chang, H. García-Molina, C. Lagoze, A. Paepcke: STARTS - Stanford Protocol Proposal for Internet Retrieval and Search. Computer Systems Laboratory, Stanford University, Stanford (CA).
- [Halasz 88] F. G. Halasz: Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM*, 31(7), 1988, 836-852.
- [Hawking 99a] D. Hawking, P. Thistlewaite: Methods for Information Server Selection. *ACM Transactions on Information Systems*, 17(1), 1999, 40-76.
- [Hawking 99b] D. Hawking, N. Craswell, P. Thistlewaite, D. Harman: Results and Challenges in Web Search Evaluation. Proceedings WWW8, Toronto (ON), 1999, 243-252.

- [Kahle 93] B. Kahle, H. Morris, J. Goldman, T. Erickson, J. Curran: Interfaces for Distributed Systems of Information Servers. *Journal of the American Society for Information Science*, 44(8), 1993, 453-485.
- [Kleinberg 98] J. Kleinberg: Authoritative Sources in a Hyperlinked Environment. *Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms*, January 1998, 668-677.
- [Kwok 95] K. L. Kwok, L. Grunfeld, D. D. Lewis: TREC-3 Ad-hoc, Routing Retrieval and Thresholding Experiments using PIRCS. *Proceedings of TREC'3*, Gaithersburg (MD), NIST Publication #500-225, 1995, 247-255.
- [Lawrence 99] S. Lawrence, C. Lee Giles: Accessibility of Information on the Web. *Nature* 400 (6740), 8th July 1999, 107-110.
- [Le Calvé 99] A. Le Calvé, J. Savoy: Database Merging Strategy based on Logistic Regression. *Information Processing & Management*, 1999, to appear.
- [Leighton 99] H. V. Leighton, J. Srivastava: First 20 Precision among World Wide Web Search Services (Search Engines). *Journal of the American Society for Information Science*, 50(10), 1999, 870-881.
- [Marchiori 97] M. Marchiori: The Quest for Correct Information on the Web: Hyper Search Engines. *Proceedings of WWW6*, Santa Clara (CA), April 1997.
- [Moffat 95] A. Moffat, J. Zobel: Information Retrieval Systems for Large Document Collections. *Proceedings of TREC'3*, Gaithersburg (MD), NIST Publication #500-225, 1995, 85-93.
- [Picard 98] J. Picard: Modeling and Combining Evidence Provided by Document Relationships using Probability Argumentation Systems. *Proceedings of ACM-SIGIR'98*, Melbourne (Australia), 1998, 182-189.
- [Robertson 95] S. E. Robertson, S. Walker, M. M. Hancock-Beaulieu: Large Test Collection Experiments on an Operational, Interactive System: OKAPI at TREC. *Information Processing & Management*, 31(3), 1995, 345-360.
- [Salton 89] G. Salton: Automatic Text Processing, The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, Reading (MA), 1989.
- [Savoy 94] J. Savoy: A Learning Scheme for Information Retrieval in Hypertext. *Information Processing & Management*, 30(4), 1994, 515-533.
- [Savoy 96a] J. Savoy: Citation Schemes in Hypertext Information Retrieval. In *Information Retrieval and Hypertext*, M. Agosti, A. Smeaton (Eds), Kluwer, Amsterdam (NL), 1996, 99-120.
- [Savoy 96b] J. Savoy, M. Ndarugendamwo, D. Vrajitoru: Report on the TREC-4 Experiment: Combining Probabilistic and Vector-Space Schemes. *Proceedings TREC'4*, NIST publication 500-236, Gaithersburg (MD), October 1996, 537-547.
- [Savoy 97] J. Savoy: Ranking Schemes in Hybrid Boolean Systems: A New Approach. *Journal of the American Society for Information Science*, 48(3), 1997, 235-253.
- [Small 99] H. Small: Visualizing Science by Citation Mapping. *Journal of the American Society for Information Science*, 50(9), 1999, 799-813.
- [Sparck Jones 77] K. Sparck Jones, R. G. Bates: Research on Automatic Indexing 1974-1976. Technical Report, Computer Laboratory, University of Cambridge, UK.
- [Voorhees 95] E. M. Voorhees, N. K. Gupta, B. Johnson-Laird: Learning Collection Fusion Strategies. *Proceedings of the ACM-SIGIR'95*, Seattle (WA), 1995, 172-179.
- [Voorhees 96] E. M. Voorhees: Siemens TREC-4 Report: Further Experiments with Database Merging. *Proceedings TREC'4*, NIST publication 500-236, Gaithersburg (MD), 1996, 121-130.
- [Xu 98] J. Xu, J. P. Callan: Effective Retrieval with Distributed Collections. *Proceedings of the ACM-SIGIR'98*, Melbourne (Australia), 1998, 112-120.

PLIERS AT TREC8

A. MacFarlane^{1,2}, S.E. Robertson^{1,2}, J.A. McCann¹

¹ School of Informatics, City University, London EC1V OHB

² Microsoft Research Ltd, Cambridge CB2 3NH

Abstract: The use of the PLIERS text retrieval system in TREC8 experiments is described. The tracks entered for are: Ad-Hoc, Filtering (Batch and Routing) and the Web Track (Large only). We describe both retrieval efficiency and effectiveness results for all these tracks. We also describe some preliminary experiments with BM₂₅ tuning constant variation.

1. INTRODUCTION

The work described here is a continuation and expansion of last year's PLIERS entry [1] that concentrated on the VLC2 track. In TREC-8 we have entered for three main tracks: Ad-Hoc, Filtering and the Web Track. For the Filtering track we have entries for the Batch Filtering/Routing sub-tasks only and the large task in the Web Track. The main focus of our research is in the area of retrieval efficiency and we continue that theme in this paper (we accept that this focus differs from much of the other work done in TREC). However we have attempted to improve the retrieval effectiveness in our system by looking at the issue of tuning constants for BM₂₅ and the relationship between them.

The hardware used for much of these experiments is the "Cambridge Cluster" that consists of 16 nodes each with two Pentium PII-30 processors (32 processors in all), 384 MB of RAM and 9 GB of disk space on each. The nodes are connected by a fast Ethernet switch for general operation and a Myrinet (Gigabit class) switch for use by parallel programs. Also used for comparison and indexing purposes is a Pentium PII with 1 processor, 128 MB of RAM (much less than the Cluster nodes) and 9 GB of disk space. All 16 nodes of the Cluster were used in the parallel experiments, but we did not utilize all of the processors. In Indexing we used 17 processors (1 timer process and 16 indexer processes). For search we used 18 processors (1 batch client, 1 client interface and 16 leaf processes mapped). For the Filtering track we used 16 processors (mapping was 1 processor for a master Router, 15 processors for Slave Routers). The topology used is given in [1].

The structure of the paper is as follows. Details of indexing experiments and the databases/collections used are described in section 2. Initial work done on tuning constants is outlined in section 3 and a hypothesis is put forward concerning their use. The TREC-8 Ad-Hoc experiments are described in section 4. The Filtering Track experiments are described in section 5, while the Web Track experiments and some prior experiments with VLC2 data are outlined in section 6. A conclusion is given in section 7 particularly with regard to the tuning constant hypothesis.

2. INDEXING EXPERIMENTS AND DATABASE DETAILS

The text collections used for the experiments described in this paper are taken from the 5 GB Tipster collection and the 100 GB VLC2 collection. The same stop word list [2] and stemming algorithm (Lovins) used in the VLC2 experiments [1] was applied to all of the Indexed text. An SGML/HTML parser was used to identify various aspects of documents such as end of document: it was also used to remove the <SUBJECT> field in the LA Times collection (disk 5). Controlled Fields such as XX, CN, IN etc found in FT data were indexed. We record a number of different aspects of these collections. Collection data such as the number of documents in the collection, total word length of collection and total size of text is declared. For each indexing run we record aspects of the index such as the type of distribution used for Inverted file (partitioning or replication), type of Inverted file (with or without position data), Inverted file size (together with % of text) and relevant documents per collection used for training (Filtering Track only). Information on timing data includes indexing times plus various parallel processing measures such

as speedup (sequential time/parallel time), efficiency (speedup/number of processors) and load imbalance (LI) [3] (Max node time/Average node time).

2.1 AD-HOC TRACK

Inverted File Type	Machine	Time (Hours)	Speedup	Efficiency	LI	Index Size (% of Text)
Postings Only	Pentium Cluster	0.81 0.059	13.68	0.85	1.06	324 Mb (17%) 342 Mb (18%)
With Position Data	Pentium Cluster	1.04 0.064	16.17	1.01	1.06	832 Mb (43%) 851 Mb (47%)

Table 1 - Ad-Hoc Indexing Experiment Details

Table 1 show details of Ad-Hoc Indexing experiments. The Ad-Hoc text collection consists of Tipster Disk4 and Disk5 minus the Congressional record on disk4. It consists of 528,155 documents and the text size is 1,904 MB: the total word length detected was just under 270 million words. Indexing experiments on both the Cambridge Cluster and the single Pentium were done. Inverted files were partitioned on the cluster using the local build method with document identifier partitioning [4]: a method which keeps all data local during Indexing. A total of 17 processors was used to map 16 Indexer processes and 1 timer process. This is the same method used in last year's experiments [1]. Indexes with and without position data were produced. The cluster yielded good results particularly on Indexes with Position Data where the extra memory on the Cluster paid dividends: super-linear speedup and efficiency were recorded. There is a slight increase in Index size on the Cluster due to repetition of keyword records found in the type of Index used. The increases are only minor however: 0.05% for postings only files and 0.02% for files with position data.

2.2 FILTERING TRACK

DATABASE	Batch Filtering	Routing Training 1: EXTRACT	Routing Training 2: SELECT	Test Collection
<i>Index Time (hrs)</i>	0.085	0.42	0.50	0.18
<i>No Documents</i>	64,139	251,396	256,761	140,651
<i>Text Size in MB</i>	167	979	1,013	382
<i>Index Size in MB</i> <i>(% of Text)</i>	37.72 (22.3%)	158 (16%)	162 (16%)	85 (22%)
<i>Relevance Judgments (Avg)</i>	548 (10.96)	1836 (36)	1797 (35)	1276 (25.52)
<i>Collection Word Length (Million)</i>	27	138	142	60
<i>Description</i>	FT 1992: Disk 4	1/2 of Disk4/5 Minus FT1993/4	1/2 of Disk4/5 Minus FT1993/4	FT1993/4: Disk 4

Table 2 - Filtering Track Indexing Experiment Details

Table 2 shows details of Filtering Track experiments. The Indexing for all the databases was done on the single Pentium. Indexes with postings only data were produced for all the databases. The Batch Filtering database was replicated (by copying) across the Cluster that is each node in the Cluster has the same index. The Routing Training data was split into two collections: one for query extraction and another for query optimization: named EXTRACT and SELECT. Files from the Routing collection were distributed evenly to EXTRACT and SELECT, which differs from the method used in Okapi experiments of the same type [5]. Of the Routing collections only SELECT was replicated (by copying) across the Cluster (the

EXTRACT database is used to make the term pools and is not used in the term selection process). The Test collection was kept on the single Pentium.

2.5 WEB TRACK

COLLECTION	WT100g	BASE10	BASE1
<i>Index Time (hrs)</i>	3.04	0.29	0.025
<i>LI</i>	1.10	1.06	1.10
<i>Scaleup</i>	-	WT100g:0.91	WT100g:0.8 BASE10:0.87
<i>No Documents</i>	18,500k	1,870k	187k
<i>Text Size in GB</i>	100	10	1
<i>Index Size in GB</i> <i>(% of Text)</i>	10.64 (11%)	1.21 (12%)	147MB (14%)
<i>Collection Word</i> <i>Length (Million)</i>	8,600	865	87
<i>Description</i>	Full Db	10% of WT100g	1% of WT100g

Table 3 - Web Track Indexing Experiment Details

The Web Track collection (WT100g) consists for 100 GB of spidered web data and was originally used in last years VLC2 track [6]. The BASE1 and BASE10 are baseline collections of the WT100g. All three collections were distributed as evenly as possible across the 16 nodes of the Cluster by linear assignment i.e. 1st x files are placed on the 1st node, 2nd x files are placed on the 2nd node etc: x is approximately total collection files divided by the number of nodes. Inverted files were partitioned on the cluster using the local build method with document identifier partitioning [4]. A total of 17 processors was used to map 16 Indexer processes and 1 timer process. The indexing times for the Web Track collections compare favorably with the results given at last years VLC2 track: the times stated above are faster than all VLC2 indexing times and meet the standard sought at last years VLC2 (an indexing time of 10 hours or less). The load balancing for all Indexing experiments on the Web Track is good with only slight levels of imbalance recorded: this confirms that the strategy used for distributing the collection to nodes was a good one. The Index time scaleup from the baselines to WT100g and from BASE1 to BASE10 are good, with very little deterioration in time per index unit. The index sizes also compare very well with only one VLC2 participant yielding smaller Indexes than the figures we quote above. The Indexes produced on all collections contained postings only data.

3 TUNING CONSTANTS FOR BM₂₅

One aspect of PLIERS that has yet to be fully investigated is the retrieval effectiveness of the system. One way of looking effectiveness is to examine the issue of tuning constants for the weighting function BM_{25} . The method does not require much effort in order to increase effectiveness and experiments can be conducted very quickly and easily. There are two constants defined for BM_{25} [7]: **K1** that effects the influence of term frequency while the constant **B** is used to modify the effect of document length. Given that there has been no systematic work done with Okapi we also decided to examine the relationship between the two constants as well as the relationship between those constants and other variables such as Recall and Precision.

The collection used for initial experiments on tuning constants was AP 1989/90 from last years Filtering track (also used for preliminary Filtering/Router runs – see below) and the VLC2 collection also from TREC-7. We used topics' 1-50 over the AP data in with queries defined from Title Only, Title & Description and Title/Description/Narrative. For the VLC2 data we used topic 351-400 in Title Only form (we wanted the best possible figure in order to improve effectiveness for the Web Tracks small queries). The relevance judgments used were taken from TREC-7 for those topics. The values for **B** ranged from 0.1 to 0.9 as **B** is always in the range 0-1. We were more flexible with **K1** as values can go

from 0 to infinity and varied values with each query type depending on how interesting the results were. We investigated a number of different measures including TREC average Precision, Recall and Precision at 20. Our criterion however was to find the best combination of **B/K1** on average Precision criterion. We plotted **B** against **K1** and ran experiments against each other plotting the evaluation measure as a third variable. We then chose the best **K1/B** combination for a given query for use in Ad-Hoc and Web Track experiments (the constants chosen are reported in the relevant sections below).

We give a very brief description of our experimental results and some conclusions arising from them. In the case of average precision on the AP data it was found that **K1** tended to be more significant than **B** except when **B** < 0.3 with Title/Description/Narrative queries. The VLC2 data was measured with Precision at 20 with which **B** was more significant. With Recall there tended to be more of an interaction between the constants, apart from Title/Description on AP data where **K1** was dominant. It is clear from the data that there are trends and that the tuning constant variation data is not random. The data does however yield different shapes of surface depending on effectiveness measure, query content and collection used (given the small number of collections used). We would say that in general **K1** is more significant than **B**, but there does appear to be a noticeable interaction between the constants with some evaluation measures. The implication is that Term Frequency tends to be significant when using the *BM₂₅* term weighting function with some measures.

We wish to investigate the following hypothesis: **K1/B** values from one data set (where a data set is defined as a set of queries and a collection), are good predictors for retrieval effectiveness in another data set, irrespective of measurement used. We will return to the validity of this hypothesis in the conclusion.

4 AD-HOC TRACK EXPERIMENTS

The purpose of the Ad-Hoc experiments was to examine the issue of term weighting with no passages (referred to simply as term weighting in the rest of this paper) versus passage retrieval search. The passage retrieval mechanism used is from Okapi experiments at TREC [8]. We submitted five runs and recorded extra runs for the two types of search for queries derived on topics' 401-450. For passage retrieval we did one parallel run and one uniprocessor run. For Term Weighting we did two parallel runs and two sequential runs composed of runs on Indexes with and without position data. We prepared Title, Title/Description and Title/Description/Narrative queries for each topic from Okapi generated queries. The run identifiers together with their query processing type and query type are given in table 4. Extra runs were done on the single database as well in order to compare times.

TRACK RUN-ID	QP TYPE	QUERY TYPE	COMMENTS
plt8ah1	PASSAGE	Title Only	2 Timing runs
plt8ah2	PASSAGE	Title/Description	2 Timing runs
plt8ah3	PASSAGE	Title/Descr./Narr.	2 Timing runs
plt8ah4	Term W.	Title Only	4 Timing runs.
plt8ah5	Term W.	Title/Description	4 Timing runs.
plt8ah6	Term W.	Title/Descr./Narr.	4 Timing runs. Not submitted.

Table 4 - Details of Ad-Hoc Track runs

The term weighting function used for all runs was *BM₂₅*. The tuning constants for Title Only are: **K1**=2.5 and **B**=0.9. For Title and Description we used a **K1** value of 2.5 and a **B** value of 0.6. A value of 4.0 for **K1** and 0.7 for **B** was used for Title/Description/Narrative queries. The choice of this tuning constant data was based on experiments described in section 3. The average length of the queries was: 2.42 for Title Only, 9.88 for Title/Description and 24.74 for Title/Description/Narrative.

4.5 Retrieval Effectiveness

Table 5 shows our Ad-Hoc results for TREC8. We include in the table the results for submitted runs, results for revised runs after a bug (in the parallel program) was fixed and the best values found by varying the tuning constants **K1** and **B**.

TRACK RUN-ID	SUBMIT EVALS	REVISED EVALS	BEST (CONST VALS)
plt8ah1	0.165	0.212	0.238 (K1=1.0,B=0.3)
plt8ah2	0.160	0.190	0.189 (K1=1.5,B=0.6)
plt8ah3	0.135	0.165	0.161 (K1=1.5,B=0.8)
plt8ah4	0.139	0.181	0.234 (K1=1.0,B=0.3)
plt8ah5	0.149	0.180	0.190 (K1=1.5,B=0.6)
plt8ah6	0.123	0.150	0.157 (K1=1.5,B=0.8)

Table 5 – Average Precision Results for AD-HOC runs

The original results submitted were very poor, but the revised results for the Title Only query using passage retrieval with the chosen tuning constants are much improved, if a little low. Results on long queries are not particularly good for any of those runs. The passage retrieval Title Only revised run (plt8ah1) produced results in which 24 out of the 50 topics were better than the median. This figure was reduced to 22 out of 50 for the revised term weighting run (plt8ah4). In both of these revised runs we record an average precision for topic 431 which was better than the best Ad-Hoc run: 0.558 compared to 0.526. It should be noted that nothing special was done with the queries such as using relevance feedback and/or thesauri.

The tuning constants chosen for all Ad-Hoc queries were not good predictors for the data used this year. In all cases with all types of query, the graph shape of the Ad-Hoc data is entirely different to that of the AP data. In the Ad-hoc data **K1** tending to be more significant for short queries while **B** tended to be more significant for longer queries. A better pair of constants was available and we show them with their results in table 5. We can state that the hypothesis declared in section 3 above is invalid when a source collection for choosing tuning constants is different from the target collection on which the constants are to be applied.

With respect to retrieval effectiveness gain of passage retrieval over term weighting we found that there were slight improvements for revised submitted runs, but for the best term weighing constant values there was little improvement and for Title/Description queries we actually recorded a slight reduction in average precision. It seems likely from this that the best tuning constants in term weighting search may not be the best pair when applied to query processing using passage retrieval: all tuning constants applied to Ad-Hoc were gathered on term weighting runs. It should be noted that the problem we had last year with the passage retrieval module was a minor bug and easily resolved when found.

4.6 Retrieval Efficiency

The response times for all 18 processor parallel runs were good with all average query processing times under 10 seconds [9]: 7 of the 9 parallel runs were under a second. The single processor runs registered good times with a maximum of 11 seconds for Title/Description/Narrative, while 7 of the 9 runs had an average query processing time of under 5 seconds.

The measurements of parallelism (speedup,efficiency, Load Imbalance [LI]) varied considerably depending on Query and Inverted file type. The most disappointing was Title Only queries on postings only data: a speedup of 4 and efficiency of 0.29 was registered with the worst LI of 1.44. It is clearly difficult to justify parallelism for queries applied to a small collection Index loaded on the “Cambridge Cluster” as communication time will dominate processing time. The best speedups were found with Title/Description/ Narrative queries on files with position data: a super linear speedup of 16.89 was recorded with an efficiency of 1.05: more memory is available in the parallel machine. LI was good in 6

out of the 9 runs with recorded measurements of 1.06 or under. With passage retrieval the parallel version examines 16 times more documents (16,000 as against 1,000) than the sequential version. All the passage retrieval runs managed to reduce the average query processing time and still examine the extra data: the best example was Title Only that took 0.83 seconds on average (sequential run) as against 0.36 on average for the parallel run.

There are a number of extra costs involved in utilizing Passage Retrieval and the position data which is recorded in Inverted files, that is costs of searching an Index with postings only data as compared with searching on one that contains position data. With respect to the extra costs of Passage Processing, there was very little noticeable difference on the sequential runs (largely because of main memory restrictions on the data used). However there was a marked change comparing search times on the parallel processing runs with much larger times for Passage retrieval over ordinary term weighting search: the worst being a factor of 18 when using Title/Description/ Narrative queries. The extra cost on Term Weighing search on Indexes with position data as against Indexes without position data was found to be consistent with a factor just over 2.2 for sequential runs, while the extra cost on the parallel runs was 1.5. The extra I/O bandwidth that is deployed with this type of parallelism (Shared Nothing) yields benefits when comparing the search performance on Indexes with postings only data and those containing position data. It should be noted that the figures declared in this section and used for comparison are optimistic given that each node in the "Cambridge Cluster" has three times the memory of the uniprocessor Pentium used.

5 FILTERING TRACK EXPERIMENTS

TRACK RUN-ID	SUB-TRACK	ALGORITHM	OPERATION
plt8f1	BATCH FILT.	FIND BEST	ADD/REMOVE
plt8f2	BATCH FILT.	FIND BEST	ADD/REWEIGHT
plt8r1	ROUTING	FIND BEST	ADD/REMOVE
plt8r2	ROUTING	FIND BEST	ADD/REWEIGHT

Table 6 - Details of Filtering Track runs

The main purpose of the Filtering track experiments is to show speed efficiency for the term selection algorithms used in the past for Filtering/Routing by Okapi at TREC [5]. Recall that 16 processors were used in these experiments. For the purpose of these experiments we used the Find Best algorithm: this is largely because earlier experiments (to be published later [11]) indicated that the best time reduction/effectiveness is achieved with that method of term selection. In this paper we report only on runs done on 16 nodes using 16 processors (other runs will be reported in [11]). Topics without relevant documents were not treated differently from topics with relevant documents as we were testing the parallelization of the query optimization algorithms. We used two operations in conjunction with the Find Best algorithm: Add/Remove terms and Add terms with re-weighting. They are both more computationally intensive than Add only and yields better retrieval effectiveness. We treated the databases differently in the different sub-tracks. With Batch Filtering runs we did extraction of terms and term selection on one database: this was because of the small number of relevant documents available in the training set. However with Routing we were able to do extraction of terms on one database and term selection on another as per Okapi experiments [5]. The number of relevant documents in the Routing training set allowed us this flexibility (the main reason for splitting the training set when using the term selection algorithms is to reduce the overall level of overfitting). All term selection runs were optimized using TREC average Precision: we tried using the utility function LF1 but the results were poor. This confirms the Okapi experiments result which predict that average Precision is a good predictor for other measures, but the other measures do not predict each other well. All Batch Filtering runs were optimized for the U1 utility function.

We did some initial experiments with last year's AP Filtering track data to test the software: the results are reported briefly in the retrieval effectiveness sections below. We split the discussion of Filtering

track experiments into two sections one of effectiveness and one for efficiency. Each section has discussion on the sub-tracks entered: Batch Filtering and Routing. We entered four runs for TREC 8: details of these can be found in table 6.

5.1 Retrieval Effectiveness

a) Batch Filtering

TRACK RUN-ID	SELECTION (Recall/Prec)	TEST DB (Recall/Prec)	AVERAGE EVALUATIONS PER TOPIC	AVG SCALE D LF1
plt8f1	0.856/0.843	0.142/0.280	907	0.354
plt8f2	0.855/0.849	0.149/0.287	2022	0.376
AP Run	0.852/0.816	0.250/0.118	5764	-

Table 7 - Details of Batch Filtering Efficiency Results

Both submitted filtering runs were optimized for the LF1 utility function. We present the results in table 7: the average scaled utility function used is from Hull [10]. The Recall/Precision for the selection runs are all very good indeed: the number of relevant documents per topic is 51 compared with just under 11 per topic for the FT data therefore our runs this year have done better with less data. This is also true of Precision on the FT test database but not true of Recall. Our filtering runs for this year sacrifice Recall for Precision. The Precision for filtering is comparable with Routing results (see table 8 below) and much higher than was expected given the type of method used for filtering and the number of relevant documents available. Comparing Add/Remove operation to Reweight we found an increase of 2.5% for the former over the latter: this increase is not particularly significant given that Reweighting need 2.2 times the Average evaluations per topic than Add/Remove. The increase in scaled average utility was more significant: using Reweight operation yielded a 6% advantage over Add/Remove.

b) Routing

TRACK RUN-ID	SELECTION (Recall/Prec)	TEST DB (Recall/Prec)	AVG EVALS PER TOPIC
plt8r1	0.873/0.696	0.858/0.286	1932
plt8r2	0.887/0.734	0.845/0.288	5364
AP Run	0.824/0.608	0.543/0.286	5481

Table 8 - Details of Routing Efficiency Results

As with Batch Filtering we did one test run on AP data with Find Best using the Add re-weight operation. The Precision/Recall for term selection was high with values of 0.824 and 0.608 respectively. The results on the test collection compared favorably with participants of the TREC-7 Routing sub-track: 5 of the 10 runs submitted produced a better average Precision of 0.286. Recall/Precision for this years runs on selection data was very good indeed with Recall just under 0.9 and precision around 0.7. Results on the test database are very good on recall which is about 0.85, while precision was adequate at around 0.28. Comparing Reweight operation as against Add/Remove we found that Reweight did bring benefits over Add/Remove but the gain was only 0.7%. This figure is not much of an increase for the extra work needed in Reweight where a factor of 2.78 more evaluations were needed over Add/Remove. With respect to precision on run plt8r1, 20/50 topics were better than median while two equaled the best: topics 355 and 380. For run plt8r2 15/50 were better than median and the same two topics as in plt8r1 equaled the best. It should be noted however that these two best yielding precision topics only contained one relevant document each. In two of the topics 387 and 394 run plt8r2 recorded the best average precision. Overall

the results were acceptable, if a little disappointing compared with other participants in the Routing sub-track.

5.2 Retrieval Efficiency

a) Batch Filtering

The average query selection time for the AP data set was 115 seconds, taking on average 26.56 iterations to select an average of 28.2 terms. The results submitted on the FT data set for TREC-8 are in stark contrast. For run plt8f1 the average term selection time per topic was 6.7 seconds with an average of 8 iterations choosing an average of 9.3 terms. Run plt8f2 was slightly more costly computationally taking 19 seconds per topic on 8.5 iterations with an average of 10 terms chosen per topic. The LI was very poor for FT data: a LI of 1.65 was recorded for plt8f1 while for plt8f2 the figure was 2.15. The LI for AP data was 1.46, an improvement on the FT data figures but are still not particularly good. The reason for the reduced load balance in these experiments is that some nodes had terms which were far more costly to evaluate than others: even though the processes were given virtually the same number of terms to inspect. In the context of time it would not therefore seem to be any use in applying parallelism to the Okapi term selection algorithms for smaller databases where there are only a limited set of relevance judgments. It is important to try and find the accumulation level for relevant documents on topics where parallelism could be applied to the term selection algorithms usefully.

b) Routing

The average term selection time for AP data was 23 minutes with an average of 49.5 iterations choosing 51.5 terms on average. The average term selection times for FT data were much smaller: taking 1.75 minutes for run plt8r1 and 6 minutes for plt8r2. The number of terms chosen was an average of 21 for plt8r1 and 27 for plt8r2. The number of iterations on FT data was also much reduced being on average 20 for plt8r1 and 25 for plt8r2. The LI for add with re-weight operation was much better than for Add/Remove operation: term selection on AP data yielded a LI on 1.25 while for plt8r2 the figure was 1.28. Run plt8r1 yielded a LI of 1.33 by contrast. The results with respect to efficiency are far superior in Routing than batch Filtering: this is largely due to the size of the data set used and the number of relevance judgments available in the Routing task compared with Filtering. The size of the data set used has a considerable impact on load imbalance. The size of the collection would be therefore a factor when examining the viability of deploying term selection algorithms. Results from runs which use lesser numbers of processors will be reported later [11].

6. LARGE WEB TRACK EXPERIMENTS

TRACK RUN-ID	QP TYPE	DATABASE	COMMENTS
plt8wt1	Term W.	WT100g	1 Timing run
plt8wt2	Term W.	BASE10	1 Timing run
plt8wt3	Term W.	BASE1	1 Timing run

Table 9 – Details of Web Track runs

The purpose of our Web Track experiments was to examine the scalability of the PLIERS data structures and algorithms as well as contribute to the debate on centralized versus distributed web search indexes. We submitted three runs, one for the full WT100g and the other two for the baselines. We ran all 10,000 of chosen web queries against each of the databases, using the same query configuration for each run using 18 processors of the Cambridge Cluster. We used term weighting search with the BM25 weighting function. Details of the Web Track runs are given in table 9. We also report details of our preparatory VLC2 experiments.

6.1 Retrieval Effectiveness

a) VLC2 Experiments

QUERY GEN.	BASE1	BASE10	VLC2	QUERY SIZE
Title	0.102 (0.130)	0.235 (0.264)	0.318 (0.377)	2.46
Title/Descr.	0.117	0.256	0.370	9.46
Title/Descr/Narr	0.103	0.228	0.392	26.54
Okapi VLC2	0.111	0.240	0.429	19.34

Table 10 – Precision at 20 for VLC2 Experiments

We present the retrieval effectiveness results on the VLC2 data in table 10. The figures in brackets are evaluations with tuning constants set as: $K1=1.5$ and $B=0.2$. These were found to be the best combination on VLC2 data evaluations. The other runs are with tuning constants set as $K1=2.0$ and $B=0.6$: these settings were used in last years VLC2 experiments [1]. We declare runs on generated queries based on the Title, Description and Narrative as wells at the Okapi generated VLC2 Queries we used last year. Queries were generated from topics 351 to 400.

We have managed to improve the retrieval effectiveness of PLIERS considerably since last year's entry [1]. For example comparing the results for PLIERS at VLC2 with the Okapi VLC2 queries used we have a figure of 0.111 compared with 0.08 and 0.056: an improvement of 39% and 98% respectively. Examining the tuning constant data has allowed us to improve our Title Only queries quite considerably (we did not do experiments on other query type because of time constraints). Our Title Only results compare favorably with last years VLC2 runs where 0.377 is higher than 4 out of 18 submitted runs (which were based on tile/description in the main). However our Okapi VLC2 query runs results are only higher than 5 out of the 18 submitted runs and we therefore need to examine the issue of effectiveness on larger queries (including experiments with tuning constant variation). As with other participants in last year's VLC2 track, we also recorded a significant rise in precision at 20 moving from the baselines to the full collection [6].

b) Web Track Experiments

DATABASE	MODIFIED AV PREC	PREC @ 10	PREC @ 20
BASE1	0.189	0.320	0.269
BASE10	0.323	0.476	0.436
WT100g	0.458	0.550	0.561

Table 11 – Large Web Track Retrieval Effectiveness Results

The results from the 50 evaluated Web Track queries are very good indeed on all collections, particularly the full 100 Gbyte collection (see table 11). They represent a vast improvement over last years results and clearly reinforce the evidence provided by the Ad-Hoc experiments that effectiveness on short queries is good. The most important aspect of these experiments is that the tuning constants chosen in Title Only VLC2 runs were good predictors for retrieval effectiveness for Precision at 20: the chosen $K1$ and B values were best in the 50 evaluated Web Track runs. The trend in both tuning constant data sets is very much the same. We can clearly state from the evidence provided with the data used in the Web track experiments that the hypothesis stated in section 3 holds when the source and target collection are the same. This demonstrates the possibility of examining statistics from a given collection (and perhaps using some form of heuristic process) in order to choose a given pair of tuning constants for incoming queries.

6.2 Retrieval Efficiency

a) VLC2 Experiments

In Table 12 we present the average query processing times in seconds together with the ratios to the appropriate baseline measures. The ratio is defined as: Big Collection Response time/Little Collection Response Time. We also give a measure of Scalability that is the proportion of time as against the database or collection size: The equation is as follows:

$$\text{Scalability} = \frac{\text{Average Query Response Time (Smaller Collection)}}{\text{Average Query Response Time (Larger Collection)}} * \frac{\text{Data Size (Larger Collection)}}{\text{Data Size (Smaller Collection)}}$$

Equation 1 – Scalability Measurement

This metric has the advantage over simple ratios in that its result actually relates query processing times to the size of data in question. We give two figures in our results for comparison purpose; one relating to the actual text size and one to the Inverted file size.

QUERY GEN.	BASE1	BASE10	WT100g
Title	0.052	0.074 (1.4)	0.87 (16.8, 11.78)
Title/Descr.	0.063	0.339 (5.4)	4.4 (69.8, 12.99)
Title/Descr/Narr	0.11	1.06 (10.05)	14.64 (138.2, 13.76)
Okapi VLC2	0.14	0.93 (6.83)	12.21 (89.9, 13.2)

Table 12 – Average Query Processing times for VLC2 experiments (ratios to baselines)

The processing speed for most runs is very good. We report the average of the two runs submitted for each database. Only two of the VLC2 runs exceed the 10 second requirement for query response times, but these are the two largest queries applied to the index (see table 10 for query sizes). One of the runs, namely Title Only meets the VLC2 requirement of a 2 second or less response time for queries over the full collection [6] and compares favorably to query response times for VLC2 participants. Query processing times in proportion to the collection size showed sub-linear growth when comparing BASE1 to the other collection runs: the exception was the larger Title/Description/Narrative runs. The ratio from BASE10 to WT100g shows super-linear growth for response times for all runs.

QUERY GEN.	BS1-BS10		BS1-100g		BS10-100g	
	INV	TXT	INV	TXT	INV	TXT
Title	5.953	7.03	4.43	5.76	0.744	0.819
Title/Descr.	1.581	1.867	1.07	1.39	0.675	0.743
Okapi VLC2	1.239	1.464	0.826	1.07	0.670	0.733
Title/Descr/Narr	0.843	0.996	0.537	0.698	0.637	0.701

Table 13 – Scalability Results for VLC2 experiments

The Scalability results are given in table 13. The best scalability results are found with Title Only queries, some of which are very spectacular for both text and index Scalability. It is clear from the data that there is a strong correlation between query size and scalability: as we increase the size of the query, scalability declines. We have organized table 13 in descending order of Scalability so this can be clearly seen. This effect is due to memory use: more constituent query terms mean that larger numbers of sets have to be manipulated: this effect would clearly be more of a problem on uni-processor experiments.

b) Web Track Experiments

DATABASE	QP TIME (LI)	SCALE-INV		SCALE-TXT		RATIO	
		Bs1	Bs10	Bs1	Bs10	Bs1	Bs10
BASE1	0.027 (1.08)	-	-	-	-	-	-
BASE10	0.121 (1.03)	1.93	-	2.28	-	4.32	-
WT100g	1.616 (1.02)	1.27	0.655	1.65	0.721	57.9	13.4

Table – 14 Average Query Processing times (secs) for Large Web Track experiments (Scalability/Ratios)

The processing speed for all runs is also very good, but not as good as for Title Only VLC2 runs. However, all runs meet the 10 second requirement for query response times. Load balance is good for all runs. The prepared average query length was found to be 2.49 terms: many terms in the queries were non-content bearing words and some queries contained none at all. A frequent query was the single term “a”. Why was the response time for Web Track queries higher than Title Only on WT100g (nearly double) despite the fact that Web Track Queries are only slightly larger than Title Only Queries? We compared the average set size of the 50 Title Only to a sample of 50 Web Track queries and found that the average set size per term for Title Only was 28K, whereas for the sample Web Track queries the average set size was 48K. The memory requirements for sets are very much larger for Web Track queries than for Title Only. We experimented with differing levels of in-core keyword on the sub-set of Web Track queries and found that we got best results with only 9% of the keyword dictionary in main memory (a document map file is kept in main memory that contains document lengths needed for BM_25). There is clearly an offset between the number of keywords and document lengths you keep in memory persistently against sets being retrieved and manipulated for Query service. Having to do I/O for list elements when weighting inverted list could reduce the search performance dramatically.

Measuring Scalability over both the text and Inverted file we found that the figures for BASE1 to WT100g were very good, while the corresponding figures from BASE10 to WT100g were acceptable. The same pattern with respect to elapsed time ratios found with VLC2 Title Only runs was found with Web Track runs: BASE1 to the other collections yielded sub-linear ratios, while BASE10 to WT100g yielded super-linear ratios. From Table 13 it can be seen that Title Only VLC2 queries yield better Scalability from BASE1 to other collections than the Web Track queries: this is another effect of the memory requirements for Web Track queries stated above.

It should be noted that the Scalability/Ratio measurements from BASE1 to the other collections should be treated with some caution, since the memory requirements and communication overheads for search times are vastly different. As with some of the smaller Ad-Hoc runs it is not clear that parallelism brings much benefit on collections of BASE1's size (see section 2.1). While these results are good there is clearly some scope for improvement, in particular the application of various query optimization techniques available [12].

7 CONCLUSION

We draw a number of conclusions from the experiments described above. From our work on tuning constants we state that a hypothesis which asserts that tuning constants from one set of experiments can be applied to another set is correct, providing the same collection is used in both experiments. We further restrict this assertion by stating that we only have evidence for the validity of the hypothesis for small 2 to 3 work queries on the Precision at 20 variable. However given that we have shown there are collection dependent variables which affect the retrieval effectiveness of the chosen tuning constants (as applied to the BM25 weighting function), we would hope that further experiments on larger queries will produce further positive evidence for the validity of our hypothesis. We would also hope that other variables such as Recall and Average precision would yield favorable results. Clearly there is scope for the investigation of factors which might affect the choice of constants. This would require a much more rigorous methodology and extensive experimental framework than we have applied in our experiments.

We need to investigate why longer Ad-Hoc queries in our system do not yield good retrieval effectiveness results. Further research into query optimization techniques for Ad-Hoc search would be fruitful: this would also require an investigation into the trade offs with respect to effectiveness and efficiency found with such techniques. In the case of the Filtering/Routing track we need more evidence on the number of relevant documents needed for choosing the type of term selection technique, for a collection of a given size.

References

- [1] A. MacFarlane, S.E. Robertson and J.A. McCann, "PLIERS at VLC2", In: Eds, E. M. Voorhees and D. K. Harman, NIST Special Publication 500-242: *Proceedings of the Seventh Text Retrieval Conference Conference, Gaithersburg, U.S.A, November 1998*, Gaithersburg: Department of Commerce, National Institute of Standards and Technology, 1999.
- [2] C. Fox, "A Stop List for General Text", SIGIR FORUM, ACM Press, Vol 24, No 4, December 1990.
- [3] D. Hawking, "The Design and Implementation of a Parallel Document Retrieval Engine", Technical Report TR-CS-95-08, Department of Computer Science, Australian National University, 1995.
- [4] A. MacFarlane, J.A. McCann, and S.E. Robertson, "PLIERS: A Parallel Information Retrieval System using MPI", In: Dongarra, J., Luque, E., Margalef, T., Eds, *Proceedings of EuroPVM/MPI'99, Barcelona, LNCS 1697*, Springer-Verlag, 1999.
- [5] M.M. Beaulieu, M. Gatford, X. Huang, S.E. Robertson, S. Walker and P. Williams, "Okapi at TREC-5", In: Ed, D.K. Harman, NIST Special Publication 500.238: *Proceedings of the Fifth Text Retrieval Conference, Gaithersburg, U.S.A, November 1996*, Gaithersburg: Department of Commerce, National Institute of Standards and Technology, 1997.
- [6] D. Hawking, N. Craswell and P. Thistlewaite, "Overview of TREC-7 Very Large Collection Track", In: Eds, E. M. Voorhees and D. K. Harman, NIST Special Publication 500-242: *Proceedings of the Seventh Text Retrieval Conference Conference, Gaithersburg, U.S.A, November 1998*, Gaithersburg: Department of Commerce, National Institute of Standards and Technology, 1999.
- [7] S.E. Robertson and K. Sparck Jones, "Simple, proven approaches to text retrieval", Cambridge University Technical Report No. TR 356, May 1997.
- [8] S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3", In: D.K. Harman, NIST Special Publication 500-226: *Proceedings of Third Text Retrieval Conference, Gaithersburg, U.S.A., November 1994*, Gaithersburg: Department of Commerce, National Institute of Standards and Technology, 1995.
- [9] W.B. Frakes, "Introduction to Information Storage and Retrieval Systems", In: W.B. Frakes, and R. Baeza-Yates, Eds. "Information Retrieval: Data Structures and Algorithms", Prentice-Hall, 1992, 1-12.
- [10] D.A. Hull, "The TREC-7 Filtering Track: Description and Analysis", In: Eds, E. M. Voorhees and D. K. Harman, NIST Special Publication 500-242: *Proceedings of the Seventh Text Retrieval Conference Conference, Gaithersburg, U.S.A, November 1998*, Gaithersburg: Department of Commerce, National Institute of Standards and Technology, 1999.
- [11] A. MacFarlane, S. Walker, S.E. Robertson and J.A. McCann, "Parallel computing for a very large search space in Routing/Filtering using Relevance Feedback", Unpublished Paper.
- [12] D. Hawking, "Efficiency/Effectiveness Trade-offs in Query Processing", SIGIR Forum, Vol 32, No 2, Fall 1998.

Optimization in CLARIT TREC-8 Adaptive Filtering

Chengxiang Zhai, Peter Jansen, Norbert Roma, Emilia Stoica, David A. Evans

CLARITECH Corporation

Abstract In this paper, we describe the system and methods used for the CLARITECH entries in the TREC-8 Filtering Track. Our focus of participation was on the adaptive filtering task, as this comes closest to actual applications. In TREC-7, we proposed, evaluated, and proved effective two algorithms for threshold setting and updating—the delivery ratio mechanism, which is used to obtain a profile threshold when no feedback has been received, and beta-gamma regulation, which is used for threshold updating. This year, we explored two ways of improving filtering performance given these our threshold-setting algorithms as a basis by (1) allowing profile-specific anytime updating and (2) optimizing the other filtering system components, in particular, the retrieval/scoring mechanism and the profile vector learning. Our results show that profile-specific frequent updating indeed improves filtering performance. In addition, they suggest that optimizing the scoring function and the term vector learning component independently leads to even further improvement, providing another indication of the effectiveness and robustness of our threshold updating mechanism.

1 Introduction

Our basic approach to filtering is a vector-space-based two-step procedure similar to the one used in many other systems: first we compute a relevance score for each document-profile pair, and then we make a binary decision to accept or reject the document based on comparison with a score threshold. When, over time, feedback information becomes available for the accepted documents for any profile, the profile term vector and threshold can be updated periodically.

A filtering system capable of performing these tasks consists of the following major components:

1. An initial profile term vector creation component
2. An initial threshold setting module
3. A scoring mechanism that computes a relevance score based on a document vector and a profile vector
4. A profile term vector updating module
5. A threshold updating module

For TREC-7, we built our filtering system by adapting a regular retrieval system. This approach has the advantage that several of the main required components are already implemented in some form. For example, the creation of an initial term vector (1) and the scoring mechanism (3) are naturally supported by any retrieval system, whereas term vector updating (4) can be achieved by means of any relevance feedback mechanism. That leaves two components to be designed and implemented, namely the two threshold-related components. One component (2) sets the initial threshold at the beginning, i.e., without any feedback information, the other (5) updates the threshold based on the collected user feedback. In TREC-7, we proposed and evaluated an algorithm for each of these: the delivery ratio algorithm for initial threshold setting and the

beta-gamma regulation algorithm for threshold updating [Zhai et al. 1999]. They proved to be very effective as reported in the evaluation of the TREC-7 filtering track [Hull 1999].

This year, we examined methods for improving the overall filtering performance given these basic threshold-setting algorithms. We considered two directions: (1) allowing profile-specific “anytime” updating; and, (2) optimizing the other two components in a filtering system: the retrieval/scoring mechanism and the profile vector learning mechanism.

One special challenge in adaptive filtering is the problem of extremely sparse data. Without reasonably frequent feedback—at least some of which must be positive—no effective learning is possible. In our TREC-8 work, we explored the possibility of improving filtering performance by performing frequent profile-specific event-driven (anytime) updating. More specifically, we wanted to allow each profile to update after some small number of new documents were accepted by that profile (2 or 4 in the official experiments). It was here, however, that the limitations imposed on us by the use of a retrieval engine became significant, and this eventually prompted us to a complete redesign of the system.

The second hypothesis we tested in TREC-8 is that we can improve the filtering performance by improving individual filtering system components independently. Somewhat simplifying the classification given above, we can view the filtering problem as consisting of three facets: (a) scoring and ranking, (b) term vector learning, and (c) threshold regulation. We were interested in seeing how techniques for optimizing the scoring function and the term vector learning module would interact with techniques for optimizing threshold learning. To test the hypothesis that the positive effects of the optimization of each individual module would outweigh and supersede any negative effects or interference, we held the beta-gamma variables constant while varying other components of the system. In particular, we varied (1) the TF formula (maximum-frequency TF normalization vs. the BM25 TF formula [Robertson et al. 1994; Robertson & Walker 1994]), which mainly affects the scoring function, and (2) the coefficients in Rocchio vector learning [Rocchio 1971] (static, fixed coefficients vs. dynamic coefficients that depend on the (growing) number of positive examples).

In the following sections, we first describe the new evaluation system we used for our TREC-8 experiments. We then describe our adaptive experiments and analyze the adaptive filtering results. In Section 4, we briefly discuss our batch filtering runs.

2 The CLARIT Adaptive Filtering Evaluation System (CAFES)

Our system for TREC-7 was essentially based on the retrieval-oriented CLARIT toolkit, which formed the core of all our past TREC efforts [Evans & Lefferts 1994; Milic-

Frayling et al. 1998; Zhai et al. 1999]. Though powerful, this toolkit did not allow for easy implementation of profile-specific updating, and offered only limited experimental flexibility in other respects.

Hence, to support the experiments we intended for the current TREC evaluation, we built an entirely new evaluation system only minimally dependent on the CLARIT toolkit—the CLARIT Adaptive Filtering Evaluation System (CAFES). Like most of CLARIT, CAFES is based on the standard vector space model, and written in C++. It consists of many independent modules that interact with each other, allowing maximum flexibility and efficiency in experiments. The algorithms that are implemented include the standard vector-space retrieval algorithms [Salton 1988], supporting various kinds of TF formulas, dot-product similarity measure, the cosine measure, and Rocchio term vector learning [Rocchio 1971], as well as our new delivery ratio and beta-gamma threshold regulation methods [Zhai et al. 1999]. On the downside, the system contains as of yet no support for the use of CLARIT constraints and subdocuments (used for example in our TREC-8 Ad Hoc submission).

The Rocchio learning algorithm is implemented with the flexibility of using different TF weighting methods. Clearly, the choice of TF may affect the centroid vector. Although it is not a-priori clear whether Rocchio learning is compatible with the BM25 TF formula, which was derived based on a probabilistic retrieval model, in practice they appear to work together very well. The Rocchio learning method makes it very easy to update a vector at any time: since we maintain a vector accumulator over time, we simply need to compute the average of the accumulator and then merge the new vector with the original profile vector. Note that, as we allow the IDF to be updated over time, the incremental Rocchio accumulator could contain the sum of document vectors weighted using different IDFs.

The system offers the option to normalize each example document vector (i.e., scale the vector to unit length) before adding it to the accumulator. This gives each example an equal weight in its contribution to the angle/direction of the centroid vector. Without the normalization long vectors dominate, which may be undesirable in some situations, such as when a document treats multiple topics.

When the selected terms are merged into the existing profile vector, the truncated vector formed by the selected terms and the existing profile vector are both first normalized to unit length, after which the new term vector can be rescaled by a parametrized coefficient (the merged weight will be the sum of the original weight (or zero, if the term does not exist in the original profile) and the new term weight multiplied by this coefficient). We provided two ways to assign this coefficient: (1) constant and uniform for all profiles or (2) dynamic, i.e., depending on the number of examples from which the terms are extracted. The dynamic formula is: $C = C_{\min} * a + (1-a) * C_{\max}$, where $a = e^{-b*N}$. The dynamic coefficient is essentially a weighted average of a minimum coefficient and a maximum coefficient in which the weight is determined by the number, N , of examples used and a parameter β that controls the “sensitivity” to N . When β is zero, $a = 1$, so $C = C_{\min}$. When β is non-zero, it determines how quickly the coefficient approaches $C = C_{\max}$ as the number of examples increases.

To estimate an initial profile threshold, we implemented the “delivery ratio” method, which was described in [Zhai et al. 1999]. The idea is to set the initial threshold in such a way that the filter can be expected to accept a given ratio or proportion of documents from the stream. The acceptance ratio, or delivery ratio, essentially serves as a substitute for a utility function in the absence of relevance judgments. The actual threshold is estimated by using a small reference corpus.

For threshold updating we implemented the beta-gamma adaptive threshold regulation method proposed in [Zhai et al. 1999]. This method selects a threshold, θ , by interpolating between an “optimal” threshold, θ_{opt} , and “zero” threshold, θ_{zero} . The optimal threshold is the threshold that yields the highest utility over the accumulated training data, given the current profile term vector. The zero-threshold is the highest threshold below the optimal threshold that gives a non-positive utility over the training data under the assumption that all documents that were rejected are non-relevant. The interpolation factor is sensitive to the number of judged examples used for computing the threshold. The actual formula is

$$\theta = \alpha\theta_{zero} + (1-\alpha)\theta_{opt}, \text{ where } \alpha = \beta + (1-\beta)e^{-\gamma N}.$$

The parameter β corresponds to a correction factor for the training score bias, and γ expresses our confidence in the optimal utility threshold based on the number of judged examples, N , to compute it.

The beta-gamma threshold regulation method requires the scoring of the training examples. Doing this for all of the examples would be very expensive, but fortunately this is not necessary. All we need are the documents with scores either above or closely below the threshold point. Thus, in CAFES, we maintain a “sample set” of training documents for precisely this purpose. Specifically, we keep track of only the documents above a reference threshold, usually a certain fraction of the regular threshold (e.g., one half). Naturally, when a threshold is updated, the reference threshold will also be updated. The effect is that the sample for threshold training consists of all the documents above an adaptive reference threshold. To control the use of sample documents, we introduced a few additional parameters. For example, we can restrict the actual sample used for threshold training to those that are not “K-documents older” than the current document, unless there are too few samples, where “too few” is defined by other parameters. This gives us the flexibility to emphasize recent documents in certain ways.

One of the most significant features of CAFES is that it allows profile-specific anytime updating. The updating time is an important parameter in profile-specific updating. In our system, a profile (both the vector and the threshold) will be updated when one of the following conditions is satisfied:

1. It has accepted enough new documents, where enough is defined by a parameter.
2. It has not been updated for a long time, where the length of time elapsed is specified by a parameter. This time elapse will be referred to as the maximum update delay (MUD).

The idea behind the first condition is to allow us to control the update timing based on the amount of feedback information, which can be measured by the number of

documents accepted for the profile. Alternatively it can be measured by the number of *relevant* documents accepted, which we will explore in the future. The idea behind the second condition is to address the problem of under-delivery (e.g., when the initial threshold is set too high), by allowing a profile to update even if it has not accepted any documents. These two conditions also allow us to move smoothly between the uniform/chunk-based updating strategy and the profile-specific strategy. If the “enough” parameter is very big, then the update timing is similar to chunk-based or uniform updating, as every profile is updated after the same interval, determined by the elapsed parameter. If the lapse is very long, on the other hand, then the updating time is determined solely by the amount of delivery for each profile, which provides profile-specific timing for updating.

Of course, the threshold must be updated whenever the term vector is updated, because the old threshold (trained using the old vector) would no longer be compatible with the new vector.

3 Adaptive Filtering Experiments

3.1 Experiment design and official submissions

Preprocessing. All documents, including the testing documents and the reference documents, and all topic descriptions are pre-indexed by all the single words occurring in noun phrases as well as all the two-word noun phrases, as recognized by the CLARIT Parser [Milic-Frayling et al. 1998]. Preprocessing only eliminated the “<PROFILE>”, “<DATE>”, “<PUB>”, and “<PAGE>” fields, so the controlled-language fields remained in the corpus and were used for indexing.

IDF Statistics. The IDF statistics are needed for term weighting, which affects both scoring accuracy and Rocchio term vector learning. Because we cannot use IDF statistics collected from the testing documents, we constructed our initial IDF statistics from all the 1991 Financial Times documents (FT91) (~14MB, 5368 documents), which are not part of the testing set. Ideally, we would like to update the IDF statistics by gradually mixing the initial statistics with the term counts in the actual documents seen over the stream, as the system sees more and more documents. However, our preliminary experiments on AP and FBIS data indicate that doing so does not seem to have as much impact on the overall filtering performance as other factors. Hence, we decided to use the initial IDF statistics throughout.

Initial profile term vector. An initial profile vector is built from the original topic description (using all fields). The profile terms are assigned TF-IDF weights. There are two different TF formulas that we tried. One is the standard maximum frequency normalized TF score (MaxNorm TF) used in the CLARIT system (i.e., $0.1 + 0.9 \times \text{TF}(t)/\text{MaxTF}(d)$)¹. The other is the BM25 proposed by Robertson and colleagues [Robertson et al. 1994; Robertson & Walker 1994]. For the BM25 TF formula, since we cannot use the average length as computed over the testing documents, we simply

set it as a parameter. In all the BM25 TF experiments, the average document length is set to 1,000 (tokens). The parameter k and b in BM25 are set to 1 and 0.5, respectively. The length for the query BM25 formula is set to 20.

Scoring algorithm. Two document profile matching formulas are considered: one is the cosine measure, and the other is simply the dot product. Since only the dot product makes sense for the BM25 TF, we always use dot product when the TF formula is BM25. When the TF formula is MaxNorm, we always use the standard cosine measure.

Initial profile threshold. We use the same initial threshold setting algorithm and the same parameter value as we used in TREC-7: the delivery ratio threshold method with a delivery ratio of 0.0005. That is, our initial threshold corresponds to accepting one out of every 2,000 documents. The initial threshold is estimated based on the FT91 data.

Threshold updating. We use the same threshold updating algorithm as we did in TREC-7: the beta-gamma threshold regulation algorithm. Beta was always set to 0.1. We chose a value of 0.1 for gamma (instead of the 0.05 we used for TREC-7), because, in our preliminary experiments with AP and FBIS data, this was found to be a more robust setting. The reason for this difference might be that, instead of using examples only from the previous chunk, we now used all the past examples for each profile for threshold updating. In other words, the meaning of N in our beta-gamma method has changed slightly. Threshold updating involves the use of all past examples up to 1,000 samples. If more than 1,000 samples are used, we forget any sample older than 30,000 documents. The sample is collected by using a second reference threshold, which is set to half of the regular threshold. The profile-specific updating time is either of two or four new accepted documents. A profile also must be updated if it has not been updated for more than 3,000 documents in the testing stream.

Term vector updating. We use Rocchio term vector learning, but only positive examples are used to expand the profile. Specifically, a centroid vector accumulator is updated whenever a profile accepts a relevant document. The terms in an example document are weighted by TF-IDF weighting. The TF formula is the same as the TF formula used in scoring. So, if BM25 is the TF formula used for scoring, then it is also used to weight the terms for the purpose of Rocchio learning. When the profile term vector is to be updated, this centroid accumulator is used to select the top 20 highest scored terms to add to the original profile. We tried two different methods for setting the coefficient: (1) fixed coefficient (0.1) and (2) dynamic coefficient ($C_{\min}=0.1$, $C_{\max}=2$, $b=0.1$) [see Section 2].

Official submissions. We submitted six runs for the adaptive filtering task, including four runs optimized for LF1 and two other runs optimized for LF2 and NF1 respectively. The parameters used in each run are shown in Table 1.

Run	TF formula	Rocchio coeff	Updating Interval
CL99afL1a	MaxNorm	Dynamic	4
CL99afL1b	BM25	Dynamic	4
CL99afL1c	BM25	Dynamic	2
CL99afL1d	BM25	Static	2
CL99afL2	BM25	Static	2
CL99afN1	BM25	Dynamic	2

Table 1. Official runs

¹ We changed the constant coefficients for this TF formula with respect to the ones used in earlier CLARIT experiments ($0.5 + 0.5 \times \text{TF}(t)/\text{MaxTF}(d)$) because we found this to work slightly better in the context of the CAFES system.

3.2 Results Analysis

3.2.1 Effect of frequent profile-specific updating

The updating time can be an important parameter in adaptive filtering. Intuitively, we want to update as frequently as possible to take advantage of any feedback information as early as possible. However, with little training information, overly frequent updates may mislead the system, as the training algorithm may show divergent behavior. In our preliminary experiments on AP and FBIS data, we compared uniform updating at every N documents with (in addition) profile-specific updating at every new accepted document. We observed that profile-specific frequent updating is most beneficial for avoiding over-delivery, and usually does not hurt the performance of good-performing topics. Thus, we adopted profile-specific updating for all our official runs, but we varied the updating time. Specifically, CL99afL1b and CL99afL1c are essentially the same, except that CL99afL1b updated profiles every four new accepted documents, while CL99afL1c updated profiles every two new accepted documents. Since the only difference is the use of different updating intervals, the comparison of these two runs will show the effect of frequent profile-specific updating. The results are shown in Table 2.

Time	LF1(Int2)	#Int2>Int	#Int2<Int4	#Int2=Int4
92	1.12	23	2	25
93	0.44	3	0	47
94	0.42	4	1	45
92-94	1.98	23	2	25

Table 2. Effect of frequent profile-specific updating

It is clear that more frequent updating consistently improves performance. We also see that the greatest difference comes from the difference at the time period 92. For periods 93 to 94, there is no difference for most topics. We suspect that this is largely the effect of stopping over-delivery.

Figure 1 shows the effect of different updating intervals (1, 2, & 4) when the MUD constraint is not used.

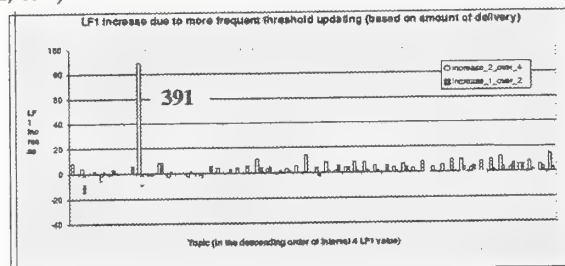


Figure 1. Effect of updating frequency on LF1

We see that a high updating frequency is generally better. Even so, the spectacular increase for topic 391 by 89 points for LF1 from an updating interval of 4 to one of 2 is unexpected. After examining the experiment trace, we found that this is because when the interval is 4, the threshold was raised to a high value quite early in the stream leading essentially to a shut-off of this profile. Specifically, at the second updating point (i.e., after accepting 8 documents), the training set contained two very high-scoring non-relevant documents, and this boosted the threshold up to a value at which it rejected almost all of the subsequent documents. With an interval of 2, the updating points were different, the slightly different threshold accepted a few more documents, and though the threshold was still high, it was low enough for the beta-gamma algorithm to be able to regulate the threshold within an appropriate range. This example shows that with frequent

updating the performance is more sensitive to the initial training examples, and that unreliable initial feedback information may cause long-term damage to the profile. It also warns us to be careful about interpreting the results, i.e., the large 'improvement' for topic 391: indeed, had we updated the profile with a much lower frequency, we would not have suffered from the accidental bias in the training sample at all!

We also looked at the effect of the Minimum Update Delay. We observed that while using the MUD seems to have a positive effect when the interval is 2 or 4, it generally has a negative effect when the interval is 1. Our original intent of using the MUD for coping with under-delivery did not work very well.

3.2.2 Effect of TF weighting

The use of different TF formulas affects the filtering performance in several ways: (1) through the scoring function, which affects score range and ranking accuracy and (2) through vector updating, as different terms might be selected or a term may receive a different weight. The difference in score range may also affect the effectiveness of threshold setting, since the parameters in our threshold method might be sensitive to the actual score range. Since we found the BM25 TF formula to be generally more effective for retrieval, we were curious whether it would also improve filtering performance. Some of our official submissions were designed to measure this effect. Specifically, CL99afL1a (MaxNorm TF) and CL99afL1b (BM25) differ only in the TF formula they use. (Note that this also implies the use of different similarity measures, as the BM25 TF formula is best used with the dot product [Robertson & Walker 94], whereas the MaxNorm TF formula is used with the cosine similarity measure.)

The results of the comparison of CL99afL1a and CL99afL1b are shown in Table 3. In the table, we have shown the number of topics for which one TF formula is better than, the same as, or worse than the other. We also show the difference of average utility.

Time Period	LF1(BM25)-LF1(MaxNorm)	#BM25>MaxNorm	#BM25<MaxNorm	#BM25=MaxNorm
92	-0.64	19	27	4
93	0.3	12	9	29
94	0.62	9	6	35
92-94	0.28	20	24	6

Table 3. Topic count comparison between MaxNorm TF and BM25 TF formulas

It appears that at the beginning (i.e., for 1992 documents), MaxNorm outperforms BM25 both by average utility and by topic count, but for the later time periods BM25 is better. Overall for all three years of documents, BM25 is better for the average LF1 utility, while MaxNorm is better for more topics. We can also see that most of the difference results in the beginning period. Indeed, for 93 and 94, there is no difference at all for a majority of topics. There are a few topics (e.g., 391) for which BM25 significantly outperforms MaxNorm in all three periods.

It would be interesting to see if the TF formula has contributed to the utility increase by producing a better ranking. In other words, we want to know if there is a correlation between the utility increase and the ranking accuracy improvement.

In Figure 2, we show, for each topic, the increase of LF1 utility value (i.e., $LF1(BM25) - LF1(MaxNorm)$) over the

whole 92—94 period, along with the scaled increase in the average precision, i.e.,

$$100 \times (\text{avg_pr}(\text{BM25}) - \text{avg_pr}(\text{MaxNorm}))$$

Some topics were excluded because their average precision was not very reliable (for those topics neither TF formula retrieved more than three relevant documents over 92—94). The average precision over all 50 topics is much better for BM25 than for MaxNorm (0.26 vs. 0.19).

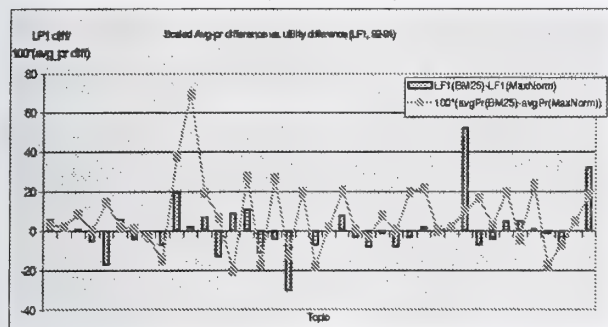


Figure 2. Ranking accuracy and utility increase

The correlation between the increase of average precision and the increase of LF1 utility is clear, but not perfect. In our system, the TF formula has a more complicated influence than just contributing to the scoring function, as it also affects the Rocchio term vector updating. Further study is needed to fully understand the effect of TF in filtering. However, it is clear that a certain minimum ranking accuracy is required if we are to expect positive utility values. If this minimum cannot be achieved it is better (in terms of utility) to return no documents at all.

3.2.3 Effect of vector updating

In our adaptive filtering model, updating the profile takes two steps. First, the term vector is updated to improve the ranking/scoring accuracy; second, the threshold is updated to optimize the decision cutoff point. Presumably, the (sparse) feedback information can benefit both vector updating and threshold updating. However, it is not a-priori clear that the score bias introduced by vector updating (because of the training documents receiving a higher score) as no negative effect on the effectiveness of threshold updating. Thus, it is interesting to evaluate whether the combination of vector and threshold updating improves the performance over either component separately.

We compared the official run CL99afL1a with an unofficial run that is the same as CL99afL1a, except that no vector updating was performed (i.e., only doing threshold updating), and found that using vector updating leads to occasionally significant changes in utility, which for some topics even turned a negative utility to a positive one. This suggests that the vector updating has indeed helped improve the ranking/scoring accuracy and thus made it possible to set a threshold to produce a positive utility.

3.2.4. Effect of dynamic Rocchio coefficient

One parameter in the Rocchio algorithm is the coefficient applied to the positive centroid vector. This coefficient controls how much weight we put on the centroid as opposed to the original term vector constructed from the topic description. Intuitively, we want to trust the centroid more if the centroid is computed using more examples. Thus, in some of our official runs, we tried a heuristic dynamic coefficient that is greater when there are more positive examples to learn from. Our official runs CL99afL1c

and CL99afL1d differ only in that CL99afL1d uses a fixed coefficient (0.1), while CL99afL1c uses a dynamic coefficient, ranging from 0.1 to 2.0. The two runs are compared in Figure 3 along with the corresponding difference in average. We can see a certain correlation between them, but there are also cases where a decrease in utility accompanies an increase in average precision, indicating a possibly harmful interaction between scoring and threshold updating.

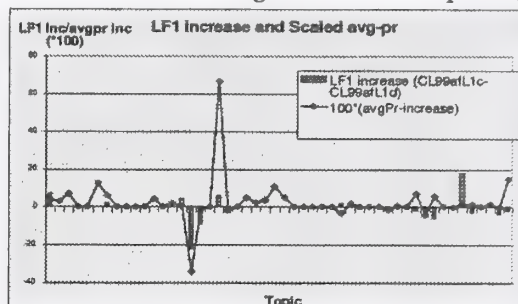


Figure 3. Utility difference vs. avg. precision difference

This effect may be due to the fact that a larger coefficient tends to increase the score bias, as, indeed, the recent training examples receive a higher weight in the centroid. As a result, the trained threshold is too high. Thus, a higher coefficient is less desirable from the point of view of threshold learning, as any benefits resulting from the use of an adaptive coefficient may be cancelled out. As evidence to support this, we observe that the average number of documents accepted for the dynamic coefficient is less than that for a fixed coefficient (7.4 vs. 10 documents/topic).

4 Batch Filtering Experiments

Our batch filtering experiments were designed mainly for validating our threshold updating mechanism. We did no preliminary experiments, but instead simply copied the parameters for threshold updating from our adaptive filtering runs. The initial threshold was trained using FT92, also with the same threshold setting parameters. To see how much value our adaptive threshold setting algorithm can add on top of initial training over FT92, we compared the batch filtering results with an alternative LF1 experiment in which no updating of either vector or threshold was performed over FT93-94. Updating turned out to be better for 16 topics and less good for 19 topics. For the remaining 15 topics there was no difference. The average LF1 utility value is better for the updating run (CL99bfL1) by 0.7.

This difference is disappointing, although, for some topics, adaptive updating does in fact significantly increase performance. For example, topic 389, exhibiting the largest increase, jumps from 83 to 122. The updating run accepted more documents for this topic than the run without updating (44 vs. 31) suggesting that our initial threshold may have been too high, and the adaptive updating helped to lower it over time.

Unfortunately, our experimental setup had a few mistakes with large consequences. For example, the runs were unable to make use of the FT92 training examples during the adaptive updating phase, and looked only at examples collected from the testing stream (FT93-94). This problem could explain the lack of benefit from adaptive filtering. In

addition, our run for LF2 used the wrong initial threshold (optimized for LF1).

Even so, some conclusions and suggestions appear warranted, based on an examination of those topics for which the system performed adequately. One observation is that our threshold setting may be generally too conservative (too high). If this is the result of score bias (artificially high score on the training documents), then increasing the value of beta in the beta-gamma algorithm should help here. Another possibility is to divide the training data in two parts, one of which is used for vector training, and the other for threshold setting. The penalty for this approach, of course, is a less effective use of training examples.

Many more analysis and experiments suggest themselves, and they are on our list for future work.

5 Summary and Further Work

We explored two approaches to improving adaptive filtering performance given the threshold setting algorithms we proposed in TREC-7. First, we tried to allow more frequent and profile-specific updating in the hope that this would result in a more efficient use of the sparse examples available in adaptive filtering. Second, we tried to improve the other two filtering system components—the scoring function and the term vector learning module—independently, with the additional aim of gaining understanding of how these two components interact with the threshold component.

Our results show that the more frequent profile updating does indeed consistently improve filtering performance. Our results also show that the improvement of scoring and term vector learning module generally leads to an improvement of filtering performance, but the relationship is more complex than can be understood from these experiments alone. One clear conclusion is that term vector updating increases the benefit of threshold updating and results in an often significant increase in utility. The effect of the TF formula is less clear: some topics are more strongly affected than others, and differences occur both in scoring and term vector learning. Finally, the term coefficients used in Rocchio learning do not seem to have a great impact on performance.

Based on these experiments and our experiments in TREC-7, we believe that our threshold setting and updating approach is effective, robust, and practically useful. In fact, our method of adaptive filtering is very general. Our approaches to establishing thresholds—both initial thresholds (using delivery-ratio threshold setting) and dynamically updated thresholds (using beta-gamma threshold learning)—could be applied in any vector-space filtering system and combined with any document-profile scoring function, as well as any term vector learning method. Moreover, our methods are also general enough to work with any utility function, including non-linear utility functions. An optimal choice of parameters may depend on the context, but we have reason to believe the general approach is robust.

In our future research, we intend to explore other possible threshold updating methods, e.g., based on logistic regression. We also hope to pay some more attention to the batch filtering task, and to study how to optimize the beta-gamma threshold regulation method in that setting.

References

- [Evans & Lefferts 1995] Evans, David A., and Robert G. Lefferts, "CLARIT-TREC-Experiments". *Information Processing and Management*, Vol. 31, No. 3, 1995, 385–395.
- [Hull 1999] Hull, David A., "The Trec-7 Filtering Track: Description and Analysis". In Voorhees, E.M., and Harman, D.K., (Editors), *The Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication 500-242. Washington, DC: U.S. Government Printing Office, 1999, 33–56.
- [Milic-Frayling et al. 1998] Milic-Frayling, Natasa, Chengxiang Zhai, Xiang Tong, Peter Jansen, and David A. Evans, "Experiments in Query Optimization, the CLARIT System TREC-6 Report". In Voorhees, E.M., and Harman, D.K. (Editors), *The Sixth Text REtrieval Conference (TREC-6)*. NIST Special Publication 500-240. Washington, DC: U.S. Government Printing Office, 1998, 415–454.
- [Robertson et al. 1994] Robertson, Steve E., et al. "Okapi at TREC3". In Harman, D.K., (Editors), *Overview of the Third Text REtrieval Conference (TREC-3)*. NIST Special Publication 500-226. Washington, DC: U.S. Government Printing Office, 1994, 109–126.
- [Robertson & Walker 1994] Robertson, Steve E., and S. Walker, "Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval". In: Croft, W.B. and van-Rijsbergen, C.J., (eds), *SIGIR'94*, Dublin, 1994, 232–241.
- [Rocchio 1971] Rocchio, J.J., "Relevance Feedback in Information Retrieval", In: Salton, Gerard (Editor), *The SMART Retrieval System*, Prentice-Hall, Englewood NJ. 1971, 313–323.
- [Salton 1988] Salton, Gerard, *Automatic Text Processing*, Addison-Wesley, Reading, MA, 1988.
- [Zhai et al. 1999] Zhai, Chengxiang, Peter Jansen, Emilia Stoica, Norbert Grot, and David A. Evans, "Threshold Calibration in CLARIT Adaptive Filtering", In Voorhees, E.M., and Harman, D.K., (Editors), *The Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication 500-242. Washington, DC: U.S. Government Printing Office, 1999, 149–156.

Filters, Webs and Answers:

The University of Iowa TREC-8 Results

David Eichmann and Padmini Srinivasan

School of Library and Information Science
University of Iowa
{david-eichmann,padmini-srinivasan}@uiowa.edu

1 – Introduction

The University of Iowa attempted three tracks this year: filtering, question answering, and Web, the latter two new for us this year. All work was based upon that done for TREC-7 [2], with our system adapted for the specifics of the QA and Web tracks.

2 – Adaptive Filtering Track

Our existing approach to search/filtering involves a dynamic clustering technique where the threshold for formation of new clusters and the threshold for visibility of ‘sufficiently important’ clusters can be specified by the user when the topic is presented to the system. The TREC requirements for multi-query support and simulation of user judgment responses led us to modify the single set-of-clusters model, creating a two-level scheme. Note that we did not use the controlled-language field in the FT database.

The primary level corresponds to the internal representation of a topic definition. The original threshold specifications were retained here to allow specification of the first-order recall of the system. We experimented with a variety of means of generating a primary similarity measure, but settled on one based upon the text of the topic's concept definitions for the submitted runs.

The secondary level is where the adaptive portion of the system functions and where we found the most benefit in parameter tuning. Each primary cluster (and hence, each topic) has a private set of zero or more secondary clusters. When a document clears the threshold for a primary cluster, it either joins an existing secondary cluster or forms a new one, based upon a membership threshold. The shift from a single membership threshold to a primary/secondary pair allowed us to achieve a tunable level of recall (by using a lower primary threshold, as mentioned above) while teasing out distinctions between candidate document clusters through use of a higher secondary threshold.

Introduction of a declaration threshold for secondary cluster similarity to the primary then gave us a means for adaptation. When a secondary cluster's similarity first exceeds the visibility threshold, its most recently added document is declared to the user and a relevance judgment is obtained. The secondary cluster is then colored appropriately. Secondary clusters containing relevant (and unjudged, if any) documents are colored green and have all subsequent members declared as relevant. Secondaries containing non-relevant (and potentially, unjudged) documents are colored red and declare no further members. A non-relevant document joining a green cluster spawns an independent and new red cluster. Adaptation then occurs over time as secondary clusters

exceed the visibility threshold and are colored, with red secondary clusters mitigating the lack of precision provided by the recall-centric primary threshold.

Secondary clusters exceeding the declaration threshold potentially contain a mix of different document types (relevant, non-relevant and unjudged). We currently address this in the following, conservative manner: if a secondary cluster contains

- the most recent document is relevant, color it green;
- the most recent document is non-relevant documents, color it red;
- fewer than a specific number (currently 10) of unjudged documents and no relevant or non-relevant documents, leave it uncolored until the first relevant or non-relevant document is added, then color it appropriately (note that this optimistic stance has a distinct effect w.r.t. false positives); and finally,
- more than a specific number of unjudged documents and no relevant or non-relevant documents, color it red (we do this pessimistically due to the low density of judged documents in the corpus).

Refinements for this year involved implementation of two primary cluster term adaptation schemes and a phrase recognizer. The first adaptation scheme supported a Rocchio-based weighting of positively and negatively judged documents in calculating the primary similarity. Due to the relatively high density of negative and unjudged documents in the document stream, negative judgments are used in the weighting only in the presence of positive judgements. Positive judgments are always used. The second, 'differential' adaptation scheme is similar to the first, except that the positive and negative term vectors are comprised only of terms not found in the other vector or in the original query vector.

The phrase recognizer loads a dictionary of phrases derived from the WordNet thesaurus and injects matched phrases into the term vectors for queries and documents as they are lexed. The original terms are retained to accommodate partial terminology matches.

We submitted two runs optimized for LF1, IOWAF992 using no phrase recognition and the Rocchio-based weighting scheme (scores shown in Figure 1) and IOWAF991 using phrase recognition and the differential-based weighting scheme (scores shown in Figure 2) The performance of the Rocchio-based approach proved to be surprisingly conservative, adapting well to negative information, but not substantially acquiring relevant documents. (Note that we do not 'turn off' queries – all topics were active for the entire run.) The performance of the differential approach is substantially the same as the Rocchio-based scheme with the exception of a small number of distinctly poorly performing topics. Our current suspicion is that this is due to the appearance in the phrase dictionary of a number of 'stop' phrases (e.g., 'and so on,' 'in point of fact,' etc.) that occur in the text of the topic, similarity was skewed higher for a number of non-relevant documents. We will be experimented with more limited phrase dictionaries as a means of controlling this, as well as with more domain-specific phrase dictionaries.

Hull, in his summary of the TREC-8 systems, computed scaled utility [3]. The scaled utility $u_s^*(S, T)$ accounts for the fact that the topics differ in the number of relevant documents that exist

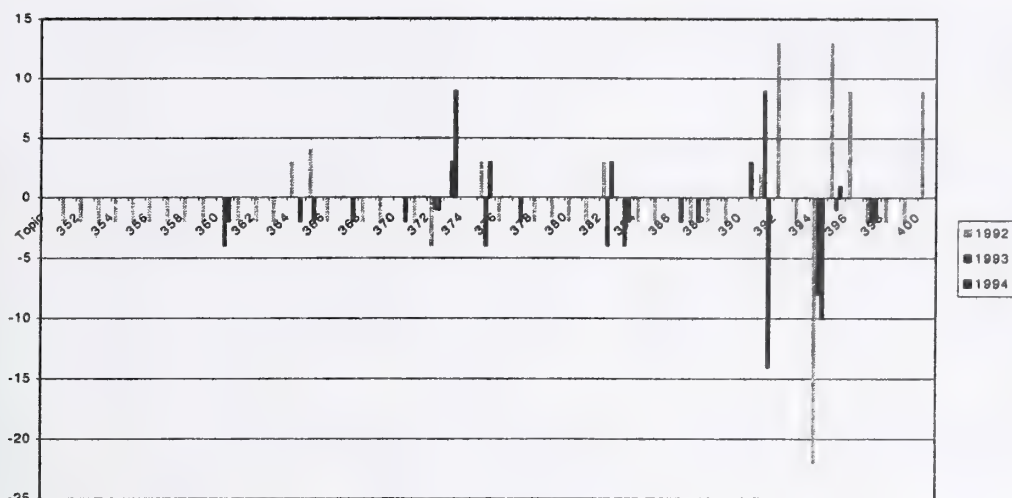


Figure 1: Rocchio Filtering - LF1

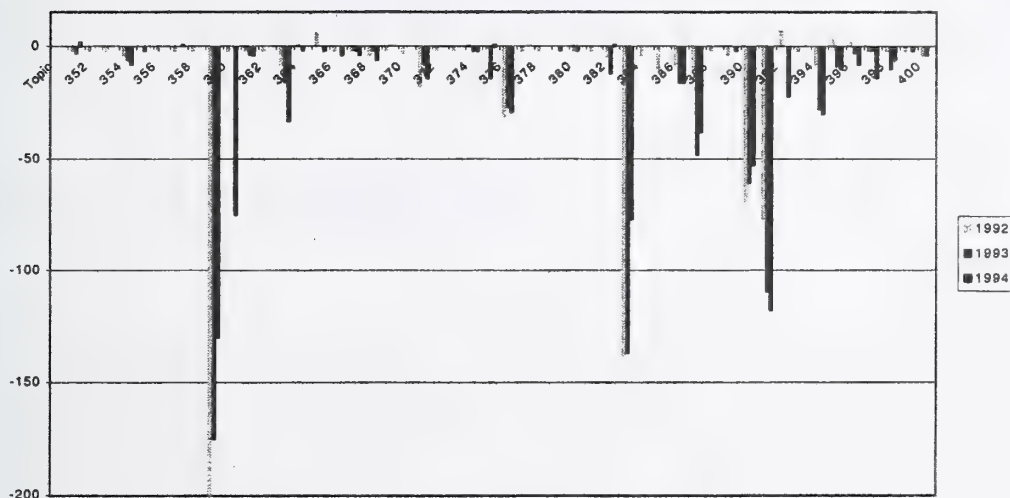


Figure 2: Differential Filtering - LF1

in the database. (The smallest number of positive judgements is 0 and the largest is 747 while the average across the queries is 114 with a standard deviation of 155).

$$u_s^*(S,T) = \{ \max(u(S,T), U(s)) - U(s) \} / \{ \max U(T) - U(s) \}$$

where $u(S,T)$ is the utility for the system S and topic T pair; $U(s)$ is the utility of retrieving s non-relevant documents (and 0 relevant ones); $\max U(T)$ is the maximum possible utility for topic T which is dependent upon the number of relevant documents present for the query.

The scaled utility normalizes performance against a given number of retrieved non-relevant documents. The scaled utility computed with $s = 25, 50, 100$ and 200 are presented for the top four TREC-8 systems in the second through fourth rows of Table 1. The table provides the number of relevant documents retrieved (RR), number of non-relevant documents retrieved (NR), number of

Group	Run Name	RR	NR	UN	LF1	$u_{25}^* -$ ubl	$u_{50}^* -$ ubl	$u_{100}^* -$ ubl	$u_{200}^* -$ ubl
	Baseline	0	a	a	0.0	0.0	0.0	0.0	0.0
U. of Iowa	Iowaf992	57	120	0	-69	-0.022	-0.012	-0.007	-0.003
Claritech	CL99afL1d	180	a	a	-100	-0.056	-0.029	-0.014	-0.007
U. of Twente	Uttnolf1f	b	a	a	-60	-0.029	-0.016	-0.008	-0.004
U. of Mass.	INQ610	266	a	a	-406	-0.113	-0.065	-0.035	-0.019

Table 1: TREC-8 Performance Scores for Top Four Systems (LF1 and Scaled Utility

a. Data is not available to us.

b. This data is not available to us. Please see the footnote on page 3. These calculations were made using the data distributed by David Hull to TREC-8 filtering track participants. The slight differences in scaled utility figures may be due to differences in rounding up strategies.

unjudged documents retrieved (UN), the LF1 scores and the scaled utility scores. (The raw data used for these calculations for the other systems are from David Hull's preliminary analyses presented at TREC-8.) *

3 – Question Answering Track

Our work in this track involved two distinct implementations employing different sources of relevance judgements.

3.1 – Run 1: Lexical Clues and Singhal's documents

Our aim was to determine how an approach based on surface analysis using lexical clues could be successful in the question answering task. We used the 200 top documents for each question distributed by Singhal. A question grammar was designed by starting with the training questions and expanding upon it using our own experience regarding the nature and structure of questions. The question grammar was used to insert appropriate tags into the free-text questions. For example, questions that began with "When" or containing the word "date(s)" or "year(s)" had a DATE tag inserted. Similarly, questions with the word "dollar(s)" or "cost(s)" had a MONEY tag inserted. In addition to DATE and MONEY, clues to tag the questions and sentences with NUMBER, and NAME were developed. These lexical clues were embedded into rules in a lex program used to preprocess the test questions. The 200 document sets were processed using a parallel method. First each document was segmented into a set of sentences. Next, each sentence was processed through an equivalent sentence grammar that also inserted the same set of tags. Finally SMART was used to retrieve the top ranking five sentences which formed the basis for the submission.

* It should be noted that when David Hull presented filtering results at the TREC-8 conference he had mentioned that the IOWAF992 run was the best. However he later provided corrected data for the filtering runs due to an error in his data for the Twente group. Our analysis of the corrected data indicates that the Twente run is slightly better than ours in that it yields -60 LF1 across all topics. However, our scaled utility scores are slightly better than theirs. All Twente data reported here are derived from the corrected data distributed by Hull.

We explored retrieval strategies based on free-text alone against retrieval based on free-text augmented with tags on the training set. The latter strategy seemed most effective. Similar explorations indicated that weighting the tags higher than the free-text terms yielded better results.

Error analysis indicate weaknesses in our sentence segmentation algorithm. Many output sentences were far from being informative. Also, there were errors in the tagging programs. For example, identifying names turned out to be very challenging. Interestingly, this simple approach yielded a mean reciprocal rank of 0.267 over the 198 questions. Answers were found in the top 5 ranks for 81 questions. When considering the difficulty of each question, this method provided the best rank for 37 questions and the second best rank for 18 questions. This covers 68% of the 81 questions answered.

In future work, we will extend our explorations with these ideas after first refining the present approach which will be followed by suitable extensions.

3.2 – Runs 2, 3 and 4: Part-of-Speech Tagging and TRECCer's documents

The remaining three runs used the top 10 documents matched out of the primary similarity scoring for TRECCer, our adaptive filtering system. Each question was tagged and matched against a coarse taxonomy of question types (basically, who/what/when/where/...) to establish the document features necessary for a match. Each matched document was segmented into distinct sentences and these sentences were then tagged using an implementation of Brill's rule-based algorithm [1]. Separate vectors of verb phrases and noun phrases were generated for each sentence and these were scored against the feature set extracted for the question. Three outputs were then generated, each at a different level of granularity. The first (*sentence*) comprised the complete sentence, truncated if necessary to fit within the 250 byte limit. Most sentences were significantly smaller than this. The second (*50 byte*) comprised the first 50 bytes of a matching sentence. The third, and most aggressive, (*noun phrase*) attempted to narrow the response down to a single clause, typically a noun phrase for 'who' or 'where' questions. A combined plot of results from the four runs appears in Figure 3. One interesting result of our two-pronged approach is that of the 23 matches at any level for the 250-byte POS approach, only 7 overlap with the matches at any level for the Lexical Clue approach.

4 – Small Web Track

Our work in the small Web track involved adapting our Web search engine to accommodate TREC-style source document specification and generation of a vector similarity score for each document against each of the queries. This is rather distinct from our normal mode of operation, where all document vectors are stored in an underlying database layer and a lexicon of term frequency is generated. Instead, due to time constraints relating to database commit overhead, we opted for an approach more akin to adaptive filtering, where the term statistics were accumulated as the run progressed. This allowed us to process the data quickly, but at the price of less-than-optimal weights for terms early in the run. Figure 4 shows the results of the content-only output. There is a definite trend as the number of relevant documents increases to fail to match a proportionally increased number of documents. This effect is due, we believe to the term frequency issue.

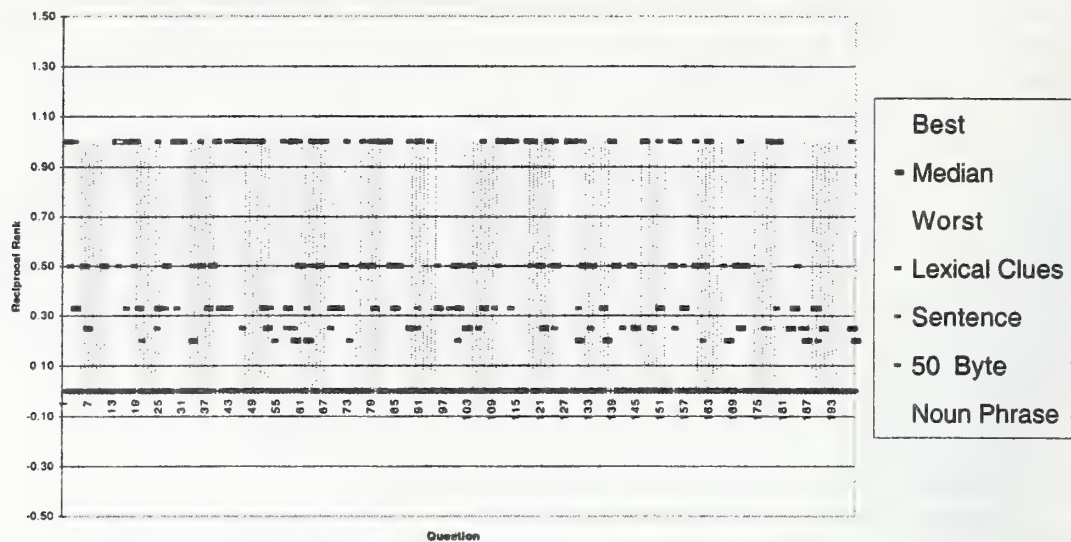


Figure 3: Question Answering Performance

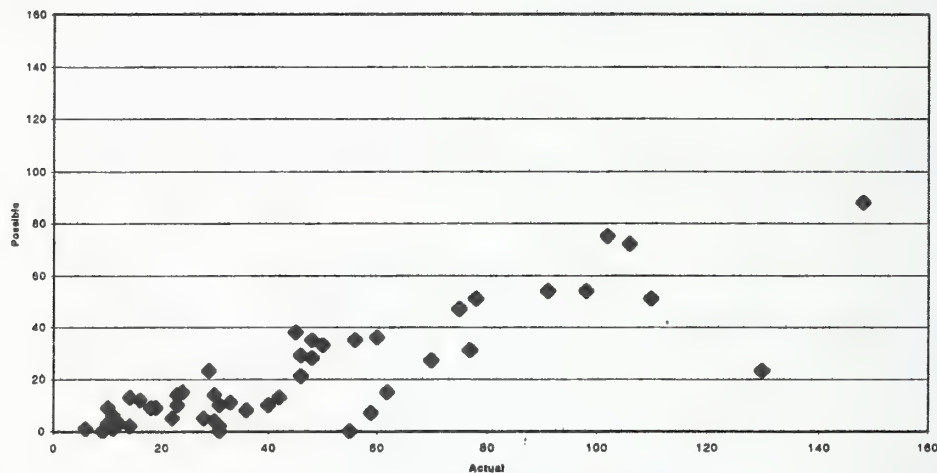


Figure 4: Web Track, Content Only

One of the hazards in the small Web track is that the sampled document are not guaranteed to comprise a connected Web subgraph. Our previous means of computing content+link scoring hence did not fare well compared to a simple content-only approach. Contrasting the exact precision against percentage retrieved of relevant documents, as shown in Figure 5, demonstrates that weighting a document's similarity with its link connectivity with few exceptions degraded performance. Because of this we feel that the small Web task, if it is to remain in the Web track, should employ documents that comprise a connected subgraph. This is much more typical of the data that would be acquired by a spider.

References

- [1] Brill, E., "A Simple Rule-Based Part-of-Speech Tagger," *Proc. of the Third Conference on Applied Natural Language Processing*, Trento, Italy, pp. 152-155.

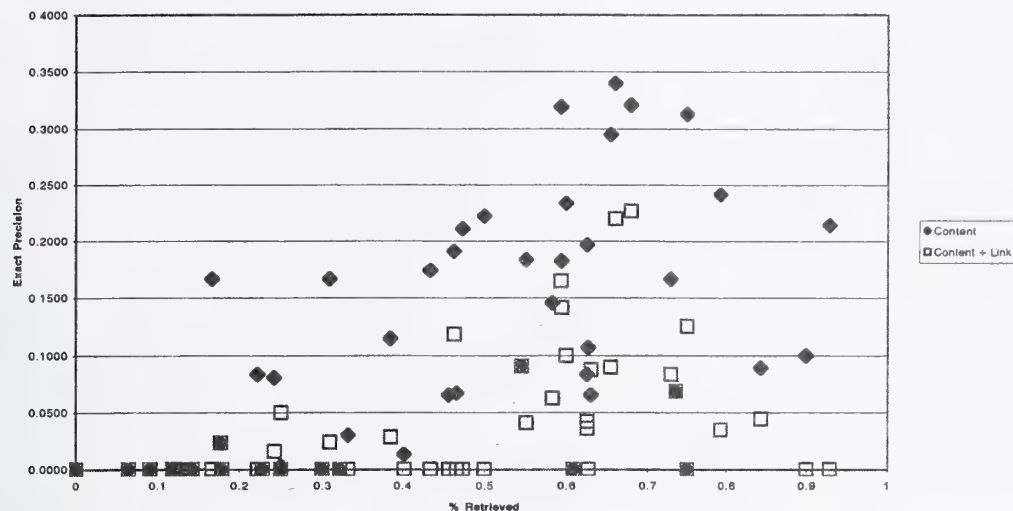


Figure 5: Web Track, Content + Link

- [2] Eichmann, D., M. E. Ruiz and P. Srinivasan, "Cluster-Based Filtering for Adaptive and Batch Tasks," *Seventh Conference on Text Retrieval*, NIST, Washington, D.C., November 11 - 13, 1998.
- [3] Hull, David. Introduction to Filtering. Text Retrieval Conference (TREC-8). November 1999.

DSO at TREC-8: A Hybrid Algorithm for the Routing Task

Hwee Tou Ng
Huey Ting Ang
Wee Meng Soon

DSO National Laboratories
20 Science Park Drive, Singapore 118230
{nhweetou, ahueytin, sweemeng}@dso.org.sg

Abstract

In this paper, we describe a new hybrid algorithm that we used for the routing task at TREC-8. The algorithm combines the use of Rocchio's formula for term selection, and an improved variant of the perceptron learning algorithm for tuning the term weights. This algorithm is able to give good performance on TREC-8 test data. We also achieved a slight improvement in average uninterpolated precision by using Dynamic Feedback Optimization (DFO) as another weight tuning algorithm and combining the ranked list generated by DFO with that of perceptron.

1 Introduction

DSO is a first-time participant in TREC. We only participated in the routing task at the TREC-8 filtering track.

Broadly speaking, there are two popular approaches to the routing task. The first approach uses the Rocchio algorithm (Rocchio, 1971), and has its root in the information retrieval community. Recently, a number of extensions have been made to this approach. These include the use of better document representation (Singhal et al., 1996), better non-relevant document selection (Singhal et al., 1997), and Dynamic Feedback Optimization (DFO) for weight tuning (Buckley and Salton, 1995). This approach has yielded very good results and is used by a number of TREC participants, including Cornell (Buckley et al., 1998), AT&T (Singhal, 1998), and NTT DATA (Nakajima et al., 1999).

The second approach treats the routing task as supervised learning from training data and has its root in the traditional machine learning community. This approach is exemplified by the work of Xerox (Schütze et al., 1995; Hearst et al., 1996; Hull et al., 1997) in the context of the routing task in TREC, as well as most of the past research on text categorization (Apte et al., 1994; Cohen and Singer, 1996; Dagan et al., 1997; Lewis et al., 1996; Ng et al., 1997; Yang, 1999). In particular, our own previous research on perceptron learning for text categorization (Ng et al., 1997) falls under this approach.

A natural question arises as to which of these two approaches is better at the routing task. One may wonder whether the success of the Rocchio formula is due to the selection of a good set of terms, or due to the assignment of a good set of weights. Although the machine learning approach for text categorization has reported good results, most of the work were only tested on the Reuters corpus (Lewis, 1992) but not on TREC data sets. Xerox has reported good results in TREC-4 routing using the machine learning approach (Hearst et al., 1996). However, their subsequent result at TREC-5 routing (Hull et al., 1997) was not as good as groups using the Rocchio approach. That this question is still unresolved is evidenced by a recent paper (Schapire et al., 1998) which attempted to compare the performance of the Rocchio approach with Adaboost, a recently developed learning algorithm from the machine learning community.

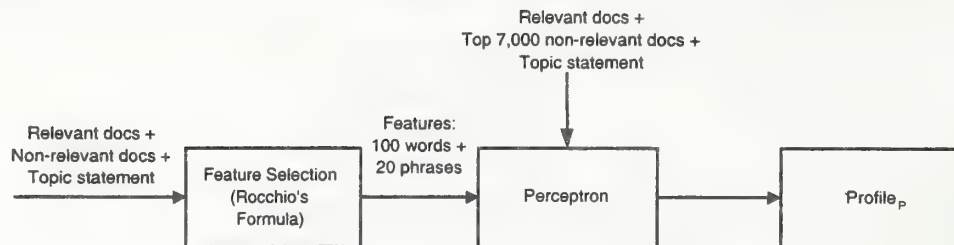


Figure 1: Training phase of our submitted run dso99rt1

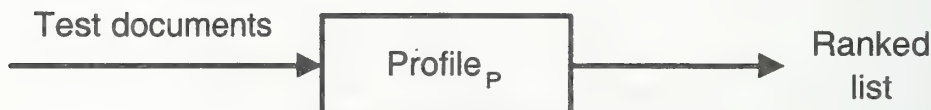


Figure 2: Test phase of our submitted run dso99rt1

In this paper, we present a new hybrid algorithm used at TREC-8 for the routing task that combines these two approaches. We treat the generation of the profile for a topic as a two-step process: first select a set of terms, and then assign appropriate weights to these selected terms. Our algorithm uses Rocchio's formula to select the terms, but after the terms are chosen, the weights assigned by Rocchio's formula are discarded. It then reverts to the use of an improved variant of the perceptron learning algorithm for tuning the weights of the selected terms. Our first submitted run dso99rt1 uses this hybrid algorithm.

To further improve accuracy, our second submitted run dso99rt2 attempts to combine two weight tuning algorithms, namely perceptron and Dynamic Feedback Optimization (DFO) (Buckley and Salton, 1995). Both submitted runs start with an identical set of terms selected by Rocchio's formula, but each algorithm separately tunes the weights and two profiles are generated per topic. The final ranked list of test documents is produced by merging the individual ranked lists of the two profiles.

2 The First Submitted Run: dso99rt1

For each topic, our routing algorithm learns a profile, which is a set of selected terms where each term is assigned a numeric weight. A term can be a word or a phrase, where a phrase is defined as any two consecutive words in a document that are both non-stop words. Once a profile is learned, it is used to rank all test documents, using the dot product score.

Our first submitted run, dso99rt1, is generated by our new hybrid routing algorithm. This algorithm consists of two parts: feature selection and weight tuning. Feature selection picks a set of terms using Rocchio's formula, whereas weight tuning is achieved by an improved variant of the perceptron learning algorithm. The broad outline of the training and test phase of our method is shown in Figure 1 and Figure 2, respectively.

Our algorithm also incorporates recent advances made in the area of document representation (Singhal,

1998; Singhal et al., 1996) and non-relevant document selection (Singhal, 1998; Singhal et al., 1997).

- Document representation: We used the document representation scheme employed in (Singhal, 1998). Each document or topic statement can be represented in *ltu*, *ltu*, or *lnu* form. These document representation forms take into account the number of times a term appears in a document, the number of documents in the training collection that contain the term, as well as a document length normalization factor.
- Non-relevant document selection: For each TREC topic, there are many non-relevant documents available for training. A non-relevant document may be one explicitly judged as non-relevant by a human assessor, or it may be considered as non-relevant based on the “complete” judgment assumption made in TREC. To be computationally tractable as well as to give high routing accuracy, it is important to select only a good subset of non-relevant documents for training. In our algorithm, there are two places where non-relevant document selection takes place. (1) During feature selection using Rocchio’s formula, the non-relevant documents are selected using the “query zone” method of (Singhal, 1998; Singhal et al., 1997). (2) In selecting the non-relevant documents for perceptron learning to tune the weights, the top 7,000 non-relevant training documents are chosen by explicitly using a learning algorithm (perceptron) to learn a classifier to rank the potential non-relevant documents.

Our TREC-8 training document collection T consists of all TREC documents (minus the 1993 and 1994 Financial Times documents) that have been explicitly judged as relevant or non-relevant to any of the topics 351-400. All documents are preprocessed where stop words and punctuation symbols are removed, and the terms are stemmed using Porter’s algorithm.¹

In the remainder of this section, we describe the feature selection and the weight tuning components of our algorithm.

2.1 Feature Selection

This component is the same as that used in (Singhal, 1998).

2.1.1 Non-relevant Document Selection

First, the topic statement is represented in *ltu* form. Then each training document in the training document collection T is represented in *lnu* form. Each training document is ranked by its dot product score with the topic statement. Let R be the set of all documents judged to be relevant to a topic i by the human assessors. Then $T - R$ is the set of all potential non-relevant documents of topic i . All non-relevant documents in $T - R$ that are ranked within the top 5,000 training documents by the dot product score, as well as all relevant training documents R of a topic are selected for use in the computation of Rocchio’s formula.

¹We were not aware of the presence of the so-called “controlled language” fields in the Financial Times documents, and so the contents of these fields are used.

2.1.2 Rocchio's Formula

Each of the selected training documents is then represented in Ltu form. The topic statement is still represented in ltu form. The following vector denoted by the Rocchio formula is then computed:

$$\alpha \times \text{topic statement vector} + \beta \times \text{average relevant vector} - \gamma \times \text{average nonrelevant vector}$$

where average relevant (non-relevant) vector is the average vector of all the selected relevant (non-relevant) training documents. In the resultant vector, we only consider the words or phrases that are present in the topic statement, or the words that occur in at least 10% of the relevant training documents, or the phrases that occur in at least 5% of the relevant training documents. We then select the top 100 words and the top 20 phrases with the highest positive weights from the set of eligible words and phrases in the resultant vector. We use the parameters $\alpha = 8$, $\beta = 64$, and $\gamma = 64$.

2.2 Weight Tuning Algorithm: Perceptron

Perceptron is the learning algorithm used in our past work on text categorization (Ng et al., 1997). Given a set of terms as features, and a set of training documents represented as feature vectors using the selected features, the perceptron algorithm can learn a set of weights that effectively discriminate the relevant from the non-relevant documents.

2.2.1 Weight Tuning

The input to the perceptron algorithm is the set of 100 words and 20 phrases selected by Rocchio's formula. However, all the weights determined by Rocchio's formula are discarded. Instead the perceptron algorithm determined from scratch the best weights of the selected terms.

All relevant training documents of a topic are used by the perceptron algorithm. In addition, 15 copies of the topic statement are added to form 15 additional relevant training documents. A subset N_2 comprising the top 7,000 non-relevant training documents are selected and used by the perceptron algorithm. The method of selecting these 7,000 non-relevant documents is described in the last part of this section. The set of training documents is represented in Lnu form. The maximum number of epochs that the perceptron algorithm iterates is 100.

We made two changes to the standard perceptron algorithm that resulted in significant improvement to the average uninterpolated precision (AUP):

1. In the standard perceptron algorithm, the final weights chosen for the selected features are those of the epoch at which the number of misclassified training documents (whether relevant or non-relevant) is minimized. However, since the number of relevant training documents in TREC is a lot less than the number of non-relevant training documents (7,000) we used, minimizing the total number of misclassified training documents tends to neglect the relevant documents. We have devised a metric to deal with the skewed distribution of relevant versus non-relevant training documents. Let R (N) be the total number of relevant (non-relevant) training documents, and let r (n) be the number of misclassified relevant (non-relevant) training documents at an epoch. We choose the feature weights of the epoch at which the metric value $r/R + n/N$ is minimized.

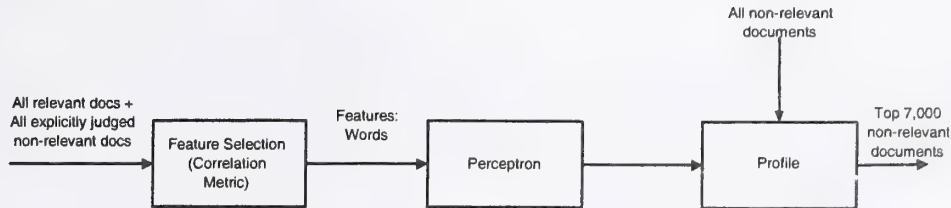


Figure 3: Selection of top 7,000 non-relevant training documents

2. Having settled on the epoch at which the weights are chosen, all terms with negative weights at this epoch are discarded before forming the final profile. This is analogous to discarding terms in the Rocchio formula with negative weights. In effect, the perceptron algorithm further prunes the set of terms chosen by the Rocchio formula, in addition to setting the weights of the remaining terms.

2.2.2 Non-relevant Document Selection

We are now left with describing the method of selecting the top 7,000 non-relevant documents to complete the description of the submitted run dso99rt1.

In our own work, we also found that the choice of the non-relevant training documents has a significant impact on the accuracy of the routing task, confirming the findings of (Singhal et al., 1997). However, instead of using the dot product score with the topic statement to rank the potential non-relevant training documents, we explicitly learned a classifier (using the perceptron algorithm) to rank the potential non-relevant documents. The approach is outlined in Figure 3.

We start with the set of all relevant documents R of a topic i , as well as the set N_1 of all documents that have been explicitly judged as non-relevant to topic i by the human assessors. Given R and N_1 , we use the correlation metric of (Ng et al., 1997) to dynamically select a set of words as features.² We first compute the correlation metric of any word which occurs more than 5 times in the relevant training documents. The average correlation metric value of all words with positive metric values is then computed. A word is selected as a feature if its correlation metric value is greater than the average.

This set of chosen words, as well as the training documents R and N_1 , are used by the perceptron algorithm to learn a profile. The maximum number of epochs that the perceptron algorithm iterates is 300. The learned profile is then used to rank all potential non-relevant training documents $T - R$, and the top 7,000 non-relevant training documents are selected to form the set N_2 of non-relevant documents mentioned earlier in this section.

²This metric has also been independently proposed by (Ballerini et al., 1997) for use in the routing task. The metric is termed U-measure in their work.

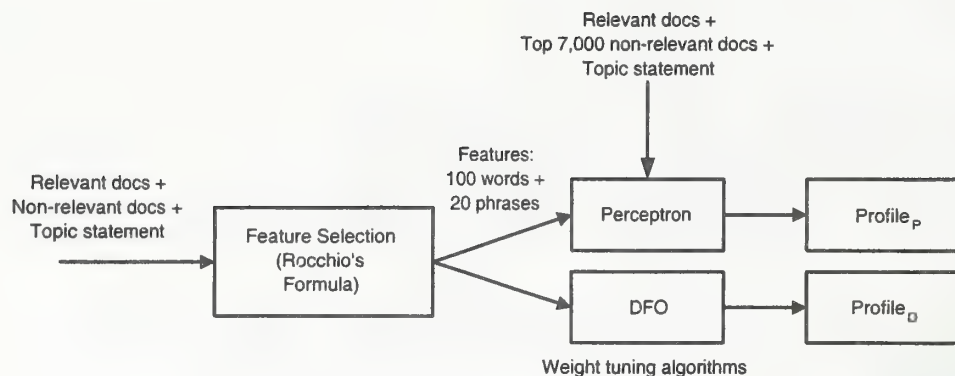


Figure 4: Training phase of our submitted run dso99rt2

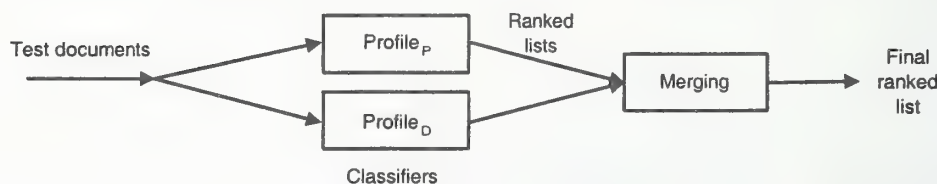


Figure 5: Test phase of our submitted run dso99rt2

3 The Second Submitted Run: dso99rt2

To further improve accuracy, our second submitted run dso99rt2 attempts to combine two weight tuning algorithms, namely perceptron and Dynamic Feedback Optimization (DFO) (Buckley and Salton, 1995). Both submitted runs start with an identical set of 100 words and 20 phrases selected by Rocchio's formula, but each algorithm separately tunes the weights and two profiles are generated per topic. The final ranked list of test documents is produced by merging the individual ranked lists of the two profiles. The broad outline of the training and test phase of our second submitted run dso99rt2 is shown in Figure 4 and Figure 5, respectively.

3.1 Weight Tuning Algorithm: Dynamic Feedback Optimization

Dynamic Feedback Optimization (DFO) has been used in conjunction with Rocchio's formula to tune the weights of selected terms in the work of (Singhal, 1998). The DFO algorithm is described in detail in (Buckley and Salton, 1995). The algorithm starts with the initial weights assigned by Rocchio's formula. It proceeds in three passes. In each pass k , the algorithm traverses the terms in ascending order of their weights. For each term, its weight is increased by a factor R_k . If the new set of weights (in which one term weight is increased by a factor of R_k) gives a higher AUP on the training documents, the new set of weights is kept, else the original set is retained. We use the increment factor $R_1 = 2.0$, $R_2 = 1.5$, and $R_3 = 1.25$.

3.2 Merging

All test documents are represented in Lnu form. Each of the two profiles generated by the two weight tuning algorithms will be used to produce a separate ranked list of the top 1,000 documents per topic. The rank of a test document is determined by the dot product score s assigned by a profile to the document.

To generate a final ranked list of top 1,000 documents, all scores assigned by each profile have to be normalized to range between 0 and 1. Let max (min) be the maximum (minimum) score assigned by a profile to the top 1,000 documents. Then the normalized score of a raw score s is $(s - min)/(max - min)$. The final score of a test document is the average of the two normalized scores from the two profiles. The final ranked list is then determined by the final score of the test documents.

4 Results

There were six groups of participants with a total of eleven submitted runs to the routing task at TREC-8 (Each group can submit up to two runs). Our two submitted runs achieved the top two scores among the eleven runs, as measured by the official metric of average uninterpolated precision (AUP).

Our two submitted runs give very close performance. The AUP score of dso99rt1 is 45.1%, while that of dso99rt2 is 46.2%. Thus, dso99rt2 is slightly better than dso99rt1, by 1.1% in AUP. Of the 48 topics with at least one relevant test document, dso99rt2 achieves scores equal to or above the median for 46 of these 48 topics. Furthermore, the maximum scores of 15 topics were contributed by dso99rt2, and the maximum scores of 11 *additional* topics were contributed by dso99rt1.

In addition, we have tested a variant of our hybrid algorithm on 4 past year TREC data sets (TREC-3, TREC-5, TREC-6, and TREC-7), and the algorithm is able to achieve very competitive scores. Our evaluation indicates that merging multiple weight tuning algorithms is able to improve the AUP score in general.

5 Acknowledgements

We would like to thank Hinrich Schütze for helpful discussions.

References

- Chidanand Apte, Fred Damerau, and Sholom M. Weiss. 1994. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, July.
- Jean-Paul Ballerini, Marco Buchel, Ruxandra Domenig, Daniel Knaus, Bojidar Mateev, Elke Mittendorf, Peter Schauble, Paraic Sheridan, and Martin Wechsler. 1997. SPIDER retrieval system at TREC-5. In *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, pages 217–228.
- Chris Buckley and Gerard Salton. 1995. Optimization of relevance feedback weights. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 351–357.

- Chris Buckley, Mandar Mitra, Janet Walz, and Claire Cardie. 1998. Using clustering and superconcepts within SMART: TREC 6. In *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*.
- William W. Cohen and Yoram Singer. 1996. Context-sensitive learning methods for text categorization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- Ido Dagan, Yael Karov, and Dan Roth. 1997. Mistake-driven learning in text categorization. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 55–63.
- Marti Hearst, Jan Pedersen, Peter Pirolli, Hinrich Schütze, Gregory Grefenstette, and David Hull. 1996. Xerox site report: Four TREC-4 tracks. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 97–119.
- David A. Hull, Gregory Grefenstette, B. Maximilian Schulze, Eric Gaussier, Hinrich Schütze, and Jan O. Pedersen. 1997. Xerox TREC-5 site report: Routing, filtering, nlp, and spanish tracks. In *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, pages 167–180.
- David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. 1996. Training algorithms for linear text classifiers. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- David Lewis. 1992. *Representation and Learning in Information Retrieval*. Ph.D. thesis, Department of Computer and Information Science, University of Massachusetts at Amherst.
- Hiroyuki Nakajima, Toru Takaki, Tsutomu Hirao, and Akira Kitauchi. 1999. NTT DATA at TREC-7: system approach for ad-hoc and filtering. In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*.
- Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. 1997. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 67–73.
- J. J. Rocchio. 1971. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System — Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, Inc., Englewood Cliffs, NJ.
- Robert E. Schapire, Yoram Singer, and Amit Singhal. 1998. Boosting and Rocchio applied to text filtering. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 215–223.
- Hinrich Schütze, David A. Hull, and Jan O. Pedersen. 1995. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- Amit Singhal, Chris Buckley, and Mandar Mitra. 1996. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29.
- Amit Singhal, Mandar Mitra, and Chris Buckley. 1997. Learning routing queries in a query zone. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25–32.
- Amit Singhal. 1998. AT&T at TREC-6. In *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*.
- Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval*, May.

Fujitsu Laboratories TREC8 Report

-Ad hoc, Small Web, and Large Web Track -

Isao Namba

Nobuyuki Igata

Computer System Laboratory Fujitsu Laboratories Ltd.
{namba,igata}@flab.fujitsu.co.jp

Abstract

This year a Fujitsu Laboratory team participated in three tracks: that is ad hoc, small web track, and large web track. As basic techniques, we compared four popular stemmers, and we made simple removing stop pattern techniques for TREC queries. For the ad hoc task, and small web track, we used the same techniques. We experimented with area weighting, co-occurrence boosting, bi-gram utilization, and reranking by bi-gram extraction from pilot search.

The effect of blind application with those techniques is rather limited, or even uncertain in the TREC8 experiment. What we can say from TREC8 result is that blind application of co-occurrence boosting and area weighting may be effective for the small web track. They require query dependent application.

In the large web track, our main interest is efficiency, that is how much resources are required to process 100GB of web text and 10000 real web queries in practical time. Using a statistical based language type checker, we can eliminate 23% of non-English text. This leads to speeding up a indexing and reducing the index size. The search speed for an inverted file is CPU intensive if the target machine has main memory in excess of 10-25% of the index size. So with simple, but effective index compression methods, the throughput of query processing is about 0.54-1.1 query/second even by a single 300MHz Ultra-sparc processor.

1 System Description

1.0.1 Teraß

Teraß[1] is a fulltext search library, designed to provide an adequate number of efficient functions for commercial service, and to provide parameter combination testing and easy extension for experiments in IR. For TREC8 we added functions for reranking pilot search results, and fixed bugs found during the TREC8 experiment.

1.0.2 trec_exec

trec_exec is designed for automatic processing of TREC. It contains a procedure controller, evaluation module, logging module, and all non-searching units such as query generation, query expansion and so on.

The motivation of making this program is that we could not fully tune the system in the TREC7 experiment. In the TREC7 experiment it took about 5 minutes to evaluate the result, and we had difficulty in analyzing what parameter was actually effective as logging of parameters was imperfect.

trec_exec can execute all the TREC processing for one run in about one minute, and it can be used for system tuning by hill-climbing.

2 Common Processing

The following process is common among all tracks.

Large web track does not use bi-gram, reranking by N-gram, and QE.

2.1 Indexing/Query Processing

2.1.1 indexing vocabulary

The indexing vocabulary consists of character strings made up of letters, numbers, and symbols, and no stop words were used in indexing. For TREC8, we modified the grammar of the token recognizer to accept acronyms with symbols such as U.S., and AT&T as one token.

2.1.2 Stemmer

We compared the popular stemmers. We selected four algorithms SMART[2], Porter¹, Lovins[4], and Pickens[5] which are popular or can be used free experimentally. Their token recognizer was modified to compare in the same condition.

The table 1 is the result of a pilot search on the same parameter settings. The parameter tuning is based on the SMART stemmer and TREC7. Best average precision was **bold face**, and worst was *italic*. T is the title field, D is the description field, and N is the narrative field. Blank field was not tested.

RUN	SMART	Porter	Lovins	Pickens
TREC4				
D	0.218,3351	<i>0.216,3447</i>	0.221,3364	
TREC5				
D	0.163,2125	0.158,2010	<i>0.149,1935</i>	
T	0.151,2050	0.146,1899	<i>0.144,1684</i>	
TD	0.159,1973	0.156,1989	<i>0.150,1823</i>	
TDN	0.208,2314	0.200,2360	<i>0.190,2233</i>	
TREC6				
T	0.211,2200	0.218,2215	<i>0.182,2120</i>	0.217,2182
D	0.173,1763		<i>0.158,1631</i>	0.170,1721
TD	0.237,2323	0.243,2320	<i>0.223,2201</i>	0.244,2324
TDN	0.254,2711	0.243,2581	<i>0.240,2624</i>	
TREC7				
T	0.194,2199	0.187,2227	<i>0.156,2129</i>	0.189,2182
D	0.210,2538	0.200,2392	<i>0.182,2329</i>	0.203,2391
TD	0.237,2323	0.215,2551	<i>0.194,2479</i>	0.219,2555
TDN	0.257,2762	0.238,2661	<i>0.230,2638</i>	0.256,2874

Table 1: Comparison of stemmers (average-precision,rel-ret)

¹We found the official home page of Porter Stemming Algorithm [3], and found that the author said almost all the implementation was different from the original one, and he made improvement of rules. Since we found the page in a few days ago, we did not try the official one.

Since the system is tuned based on the SMART Stemmer, and nothing is known about the relation between stemmer and system parameter tuning, it is difficult to deduce a concrete result.

What we feel is that SMART or Pickens are a better choice, and our choice is SMART.

2.1.3 Information in inverted file

Text number, term frequency, and term position are stored for the ad hoc task, and small web track for run time phrase processing and reranking by bi-gram extraction.

Only text number and term frequency are stored for the large web track to save disk space.

2.1.4 Stop word list for query processing

As in the TREC7[1], we used a stop word list of about 400 words of Fox[6], and words with a high df (more than 1/7 of the number of all documents) were also treated as stop words.

2.1.5 Stop pattern removal

The expression of TREC queries are artificial, so frequently appearing patterns such as "relevant document" are stop patterns. We generalized this observation, and removed the words which meet one of the following condition.

1. Word in stopword list is a stopword.
2. Word which is not a proper noun², and whose df in TREC1-7 queries is more than 400×0.1 is a stop word.
3. Word bi-gram whose df in TREC1-7 queries is more than 400×0.02 is a stop pattern.
4. Word tri-gram whose df in TREC1-7 queries is more than 400×0.01 is a stop pattern.
5. All the words in a sentence that contains "not relevant" are stop words.
6. 4 words following "other than" are stop words.
7. 4 words following "apart from" are stop words.

For the TREC-8, this parameter setting seems to remove too many patterns. The result is that official result is worse than with simple removal of stopwords about 0.5-1 point.

²U.S appears 94 times in TREC1-7 queries.

2.2 Weighting Scheme

The term weight is $qtf * tf * idf$, and the score for one document is the sum of the term weights with co-occurrence boosting.

1. qtf

qtf is the combination of the following parameters

$$qtf = \sum_f fw * tf * ttw$$

where

f is the topic field (title, description or narrative).

fw is weight of the topic field. We set the value for the title field to 3.0, the value for the description field 1.5, the value for the narrative is 0.9. Some teams [7], [8],[9] used weighting depending on field type, and we take the same approach.

tf is the bare frequency in each field.

ttw is the term type weight. It is set to 3 for terms, and set to 1 for phrase(word bi-gram).

2. tf

We simply used the tf part of OKAPI[7].

$$tf = \frac{(k_1+1)*term_freq}{(k_1((1-b) + \frac{b*doc_length_in_byte}{average_doc_length_in_byte}))}$$

$k_1 = 1.5, b = 0.75$

3. idf

We used a modified idf of OKAPI. We introduced a cut off point for low df words, and decreased the idf value for high df words.

$$idf = \log_2 \frac{N-(n*\alpha)}{n}$$

N is the number of documents
 n is df if ($df > 1/10000 * N$) else
 $n = 1/10000 * N$
 α is set to 3

2.3 Co-occurrence Boosting

As in TREC7, we use co-occurrence boosting technique which favours co-occurrence of query terms in a document. Co-occurrence boosting is implemented by simply multiplying the boost ratio to the similarity of each term.

$$S_i = \sum_t B * W_{t,i}$$

S_i is the degree of similarity between a document and topics.

i is the document number.

t is a term that document $_i$ includes.

$W_{t,i}$ is the part of similarity of term $_t$ in document $_i$.

B is the boost-ratio by term co-occurrence.

In the experiment of last year, we could not get improvement except in very short queries. One of the reasons was that applying this technique to any symbols, that is word with high idf, word with low idf, and phrase(word bi-gram) caused the theme to drift. So we limited the application of this technique only to words, and words within a limited df range.

Apparently, the best parameter B depends on the query, and we still have not found an automatic parameter control method. So we set the B to 1.10 for the title word, to 1.05 for the description word, and to 1.03 for the narrative word, and to 1.0 for the word added by query expansion.

2.4 phrase(bi-gram)

Instead of traditional IR phrase (two adjacent non-stopword pair with order or without order), we permitted limited distance in phrase. The motivation for introducing fixed distance is as follows. The first motivation is that non-stopword may exist between two adjacent words in a query. For example, in TREC7 query 351 "Falkland petroleum exploration", the word Island may be inserted between "Falkland" and "petroleum", but the two words may be located near each other in the sentence within a limited distance. The second motivation is that there are many stopword lists in the world and it is difficult to select one, and we don't like to remake the index every time we choose a different stopword set for an experiment, and stopwords are rather area dependent. The third motivation is that it is easy and fast enough to experiment by computing the frequency of the bi-gram using a word offset in an inverted file at run time.

The experimental environment is that the bi-gram uses bare idf, and its weight is 1/3 that of a normal word. The bi-gram is constructed from two adjacent non-stopword pair in a query which is not separated by stop pattern in stop pattern removal. Their source area is only the title and description fields. Table 2 shows the average precision after QE.

3 Ad hoc task

We tried many techniques in the TREC8 ad hoc experiment, and most of them did not survive as their effects were uncertain or too marginal, and were just logged by trec_exec. As the following analysis shows, except QE, most of the techniques which we applied to the official runs were severely query dependent or target document dependent.

3.1 Ad hoc official runs

The results are shown in Table 4. TREC8 query was easier than that of TREC7, and there is little difference between the title only run and title and description run.

Run-id	Flab8as	Flab8atd2	Flab8atdn	Flab8ax	Flab8at
field	TD	TD	TDN	TDN	T
Av Prec	0.290	0.293	0.324	0.316	0.287
R-Prec	0.324	0.320	0.353	0.350	0.315
P@20	0.420	0.420	0.470	0.451	0.426
Retrieved	50000	50000	50000	50000	50000
Rel-ret	3090	2990	3261	3207	3084
Relevant	4728	4728	4728	4728	4728
best	0	2	1	1	3
>= med	42	39	39	41	40

Table 4: Eleven Point Average (Official Run)

The conditions of each run are given in table 5.

Name	Flab8as	Flab8atd2	Flab8atdn	Flab8ax	Flab8at
bi-gram	+	+	+	+	+
Co-boost	+	+	+	+	+
Rerank	-	+	-	-/+	+
QE	+	+	+	+	+
Data fusion	-	-	-	+	-

Table 5: Parameter condition of official Runs

Flab8ax is data fusion of Flab8atdn and very long query version of the Flab8atd2 condition. For the TREC7 experiment, merging two different system results in 1-1.5 points up in some cases (eg. OKAPI+INQUERY). But as this result shows, it is not stable.

3.2 Ad hoc analysis

The effect of the techniques we employed, field weighting, co-occurrence boosting, bi-gram in query, and reranking by bi-gram are shown in the following tables.

bi-gram	field	boost	rerank	No QE	QE
-	-	-	-	0.243	0.271
+	-	-	-	0.247	0.288
+	+	-	-	0.251	0.290
+	+	+	-	0.252	0.291
+	+	+	+	0.241	0.293 (*official)

Table 6: The effects of each techniques (title and description)

bi-gram	field	boost	rerank	No QE	QE
-	-	-	-	0.215	0.259
+	-	-	-	0.236	0.288
+	+	-	-	0.236	0.289
+	+	+	-	0.238	0.290
+	+	+	+	0.235	0.288 (*official)

Table 7: The effects of each techniques (title)

4 Small Web Track

The processing of Small Web Track is the same as that of the ad hoc task.

No link run is submitted as we did not have enough time, and it seems difficult to use link information for 2GB text.

What we felt during system training by TREC7 queries was the searching for small web data easily drifted away from the main theme.

The result seems to support our feelings, that is field weighting and co-occurrence boosting is the most effective of our techniques, and QE is less effective than in the case of the ad hoc task.

4.1 Small Web Track official Runs

Two runs are submitted, Flab8wtdnN and Flab8wtdN. Their procedure was the same as that of Flab8atd2. Flab8wtdnN used full field, and

Flab8wtdN used title and description field. The results are shown in Table 8.

Name	Flab8wtdnN	Flab8wtdN
field	TDN	TD
link	NO	NO
Average Prec	0.340	0.332
R-Prec	0.353	0.355
P@20	0.401	0.398
Retrieved	50000	50000
Rel-ret	2279	2279
Relevant	1988	1954
best/ >= med	5/42	1/42

Table 8: Official small web track result

4.2 Small Web Track analysis

The following two tables show the combination of each technique.

bi-gram	field	boost	rerank	No QE	QE
-	-	-	-	0.289	0.325
+	-	-	-	0.296	0.323
+	+	-	-	0.304	0.338
+	+	+	-	0.315	0.340
+	+	+	+	0.301	0.332 *official

Table 9: The effects of each technique (title and description)

bi-gram	field	boost	rerank	No QE	QE
-	-	-	-	0.294	0.321
+	-	-	-	0.306	0.332
+	+	-	-	0.323	0.347
+	+	+	-	0.331	0.358
+	+	+	+	0.322	*0.346

Table 10: The effects of each technique (title description, and narrative)

* official result is 0.340.

5 Large Web Track

Our goal for the large web track is to show that a single CPU is enough for processing 100GB of Web

data, and even a slow CPU (Ultra-Sparc 300MHz) is enough. This is because inversion and searching is CPU intensive if hardware conditions are met, and such hardware are not so expensive today.

The balance between speed, precision and hardware cost was our research goal.

5.1 Hardware environment

One Sun Ultra2 workstation was used for the large web track. It has 1GB memory and has 2 Ultra-sparc 300MHz CPUs. Most of the processing was done using 1 CPU, sometimes 2 CPU were used.

5.2 islang

As pointed out by some groups[7], large web track data contains considerable non-English data. This data increases the size of the index, and slows down the inversion, and is never retrieved in English query processing.

A statistical based language type checker, called islang, which rejects both non-English text, and non-language text was originally designed to select Japanese and English text for Web search engine[10].

The basic idea of islang is that frequently appearing spelling, prefix and suffix patterns are the key expression in language type checking, and we can use the tri-gram as the pattern in English.

The algorithm is as follows.

1. Training

- L is the training corpus containing N patterns.
- The pattern is the tri-gram of case folded alphabetical ascii character.
To reflect the word construction of English, patterns in word and patterns adjacent to word boundaries are treated as two different patterns.
- Simply count the frequency of patterns and calculate the information of each patterns in the corpus.
 $n_i (i = 1, 2 \dots n)$ is the pattern in L
 f_i is the frequency for n_i respectively.
 $p_i = -\log_2(\frac{f_i}{N})$

2. Checking

- T is the target text containing M patterns.

(b) $t_i (i = 1, 2, \dots, n)$ is the pattern in T .

f_i is the frequency for t_i respectively.

(c) Calculate average information and score.

$$av = (\sum_{i=1}^{i=n} f_i * p_i \text{ for } t_i) / M$$

If t_i is not found in n_i , use $\alpha * -\log_2(1/N)$ instead of p_i

score is 2^{av}

(d) If score is over the threshold, reject text as not belonging to the language.

α is set to 1.0, and threshold is set to 40000, and TREC7 documents are used as a training corpus.

The example of rejected documents from IA001-0000-B001- set are given in Table 11.

DOCID	score	text
5	44167	Die Dapsy Dinos Leider wurde
44	81499	Sydsvenskan - Nyhetsrskt ...
52	51064630	323c 2f44 4f43 4e4f 3e0a ...

Table 11: Sample of rejected document by islang

23% of the text is rejected as non-English. The sum of word entry in inverted file is reduced to 10 million from 20million, and the index size is reduced to 4.0GB from 4.8GB without stopword condition.

5.3 Large web track result

Our main concern is the balance between performance and precision. Introducing Boolean AND condition speeds up query processing, but precision may go down. To compare simple ranking and AND condition with ranking, we submitted two runs. All runs did not use phrases, and query expansion. B+R means ranking document with AND condition of every non-stopword in a query. If the number of retrieved documents is less than 20, then ranking search is retried. This AND conditional interface is popular in actual Internet services. R means traditional accumulator method. Only fl8wlsb used index with stopwords. Table12 shows our official result.

Run-id	av-prec	P@10	P@20	Calc	speed(sec)
fl8wlsb	0.4103	0.536	0.510	B+R	0.75
fl8wlsr	0.4064	0.538	0.508	R	1.16
fl8wlsb	0.4116	0.540	0.507	B+R	0.54

Table 12: Large web official result

There is no remarkable difference in precision. B+R search and index with stopwords seems to be the best choice considering speed.

5.4 Performance of pre-processing

The preprocessing involves web detagging, running islang, and indexing. The official pre-processing data is as follows.

1. Detagging etc. took 3 or 4 weeks using 1 CPU.

The process contains gzip, gunzip, cat, copy and rm for original data due to shortage of disk space, But most of the time was consumed by a poor detagging script. After detagging text size is reduced to about 50GB.

After submission of result, we wrote C version of detagging program. Its processing time was about 10 hours including the time for gunzip the data.

2. islang takes 5.23 hours using 2 CPU.

23% of the text was rejected as non-English.

3. Indexing

It took about 30 hours using 1 CPU.

To compare the effect of the stopword, we made two indexes. 'With stopword' uses the stopword list of SMART[2], and rejects non-alphabetical symbols, digital string equal or greater than 1000000, and strings longer than 24 characters.

The official inversion is very slow because we failed to keep working area(300MB) of memory. Solaris2.6 swapped out the working area while inversion was being done. So more than half of the time was consumed by page-in and page-out. After submission of the official result, we found we could lock the work area in memory by mlock(), and mincore() system calls in Solaris2.6, but we did not retry by now(1999/10/27).

condition	time	status	work area
With stopword	30.38 hours	official	300MB
Without stopword	27.63 hours	official	300MB

Table 13: Inversion time

The Index size is given in table 14.

files	with stopword	without stopword
inverted file	3.01	4.03
dictionary	0.46	0.59
text size array	0.07	0.07
text number id	0.41	0.41
total	3.95GB	5.10GB

Table 14: Index size

5.5 Performance of query processing

In this section we show the basic figure of query processing.

5.5.1 Design of Teraß

We briefly list the features of Teraß concering performance.

1. I/O optimization
 - (a) simply locating the bitmap entry of an inverted file for the same term in continuous area.
 - (b) Clustering frequently accessed bitmap entries.(not used in TREC run)
2. Index entry is compressed by extended gamma coding [11] which aims at a balance between index size and decoding speed.
3. A Skip list [12] is used to minimize decoding cost.
4. Many code optimization techniques are used such as loop expansion, macro expansion, and other coding techniques.

5.5.2 Average Processing Speed

The regulation of a large web track says that query processing speed is the total processing time divided by the number of query. Other teams generally take the data parallel model to speed up processing, so we took the round-robin model to increase speed. The processing is very simple. We split a query file into 2 or 3 files, and run 2 or 3 search programs to the same index. 1, 2, and 3 processes are compared and the result is given in Table 15.

Run-id	1 process	2 process	3 process
fl8wlinsb	0.75	0.40	0.53
fl8wlinsr	1.16	0.75	0.87
fl8wlsb	0.54	0.39	0.63

Table 15: Query processing speed (elapsed seconds per query)

As the result shows that 2 process is the fastest in 2 CPU environment. This suggest searching is CPU intensive for some conditions, and we show the evidence in the following scitons.

5.5.3 CPU time vs real time

If CPU time is dominant in most of the searching processes, there is little difference between the distribution of CPU time and real time. The histogram of query processing speed is figure 1 . This speed only includes searching the inverted file.

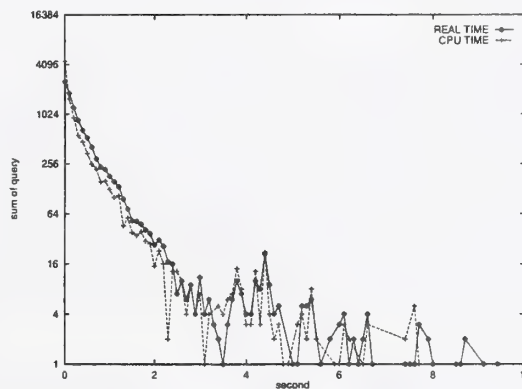


Figure 1: CPU time vs real time

For most of the queries CPU time is nearly equal to real time. But in the case of huge queries they differ.

Figure 2 shows the relation between the number of terms in query and average processing time.

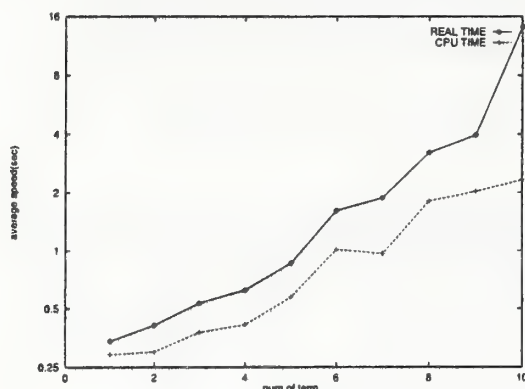


Figure 2: CPU time vs real time

The shortest CPU time is less than 1/1000 second, and the longest CPU time is 7.7 seconds. The shortest real time is less than 1/1000 second, and the longest real time is 28.1 seconds.

5.5.4 Index size vs available Memory

What actually affects the search performance is memory pages which the OS caches, and not search system caches if the index uses Unix File System⁴. But it is difficult to know what block of what files the OS caches. So we limit available main memory size using a memory resident program using `mlock()` and `mincore()`, and check search program performance.

The result is omitted in this paper, but we get the same result as that of past experiment. In our past experiments on patent abstract (index size about 1GB), the query processing was CPU intensive if memory size is more than 10-25% of index size. In our past experiment with Japanese patent data (index size 10GB), and actual query (about 13000 queries), the total of the accessed inverted file entry is about 1GB.

5.5.5 More speed

We simply list the techniques for increasing speed.

⁴This means not using raw disk

1. inversion

(a) Stemmer

The speed of the stemmer is given in Table 16.

Stemmer	speed(Gbyte/hour)
No stemmer	16.5
Porter	6.64
SMART	6.36
Lovins	7.10

Table 16: Stemmer speed comparison

Almost all stemmer implementations are run time rule matching, so they are not fast from the view of implementation.

(b) Sorting in sort merge inversion

Teraß uses a sorting merge algorithm for inversion, and quick sort algorithm is used to sort the entries of an inverted file. But the quick sort algorithm is not the fastest in this special case.

Stemming and sorting occupies more than 70% of our inversion process.

2. Searching

(a) I/O optimization

Though the main memory size is large enough compared with index size, inverted file entry which is actually accessed in query processing is not clustered.

(b) Changing Measures

We used OKAPI for all TREC8 runs. The tf calculation of OKAPI requires division every time as its form is $tf/(tf + \alpha)$. It slows down the searching speed. To the contrary, the vector space model measures such as `dtm.dnb` [13] require division only once for each document. Table 17 shows the CPU time of OKAPI and `dtm.dnb` weighting.

Measure	CPU	real
OKAPI	0.353	0.496
ddb.ddn	0.270	0.410

Table 17: Measure speed (elapsed second per query)

6 Conclusion

Though the combination of above techniques seems to be working for automatic ad hoc, and small web track, each technique is serverly query/search result dependent. We need control method whether we apply those techniques or not. The result on the Large Web track is good. The balance of indexing speed, index size, searching speed, and precision is satisfactory, considering the hardware resources, which is not measured by price because current PC is faster than old workstation, and is cheaper.

References

- [1] I Namba, N Igata, H Horai, K Nitta, and K Matsui. Fujitsu laboratories trec7 report. *The Seventh Text REtrieval Conference*, 1999.
- [2] SMART
ftp cite. <ftp://ftp.cs.cornell.edu/pub/smart/>. 1999.
- [3] Martin Porter. Official homge page of porter stemming algorithm.
<http://www.muscat.com/~martin/stem.html>, 1999.
- [4] MG Home Page.
<http://www.mds.rmit.edu.au/mg/welcome.html>. 1999.
- [5] Jeremy Pickens. Stemming and cooccurrence on a larger corpus. <http://ciir.cs.umass.edu/>, 1999.
- [6] Chiristopher Fox. Chapter 7, lexical analysis and stoplists. *Information Retrieval Data Structure and Algorithms ed. William B. Frakes, Ricardo Baeza-Yates Prentice Hall*, 1992.
- [7] S E Robertson, S Walker, and M Beaulieu. Okapi at trec-7. *The Seventh Text REtrieval Conference*, 1999.
- [8] D R H Miller, T Leek, and R M Schwarts. Bbn at trec-7. *The Seventh Text REtrieval Conference*, 1999.
- [9] James Allan, Jamie Callan, Mark Sanderson, Jinxi Xu, and Steven Wegmann. Inquiry and trec-7. *The Seventh Text REtrieval Conference*, 1999.
- [10] InfoNavigator. <http://infonavi.infoweb.or.jp/>. 1999.
- [11] Kunio Matsui, Isao Namba, and Nobuyuki Igata. High-speed text search engine (in japanese). *IPSJ 97-DD-7-3 pp15-21*, 1997.
- [12] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. Managing gigabyte - compressing and indexing documents and images. *Van Nostard Reinhold New York*, 1994.
- [13] Amit Singhal, John Choi, Donald Hindle, David D. Lewis, and Fernado Pereia. At&t at trec7. *The Seventh Text REtrieval Conference*, 1999.

	bi-gram (dist=4)	phrase(no-order)
trec4 d	0.285	0.278
trec5 d	0.192	0.186
trec6 t	0.260	0.260
trec6 d	0.185	0.187
trec6 td	0.261	0.253
trec7 t	0.244	0.235
trec7 d	0.273	0.273
trec7 td	0.280	0.281
trec8 t	0.288	0.283
trec8 d	0.256	0.257
trec8 td	0.293	0.290

Table 2: Short distance bi-gram vs phrase

2.5 Reranking by best bi-gram

The bi-gram in query is limited to non-stopword pairs. This is because any bi-gram combination of terms in query easily drifts away from its original theme with its strong idf.³ But in some cases, a non-adjacent pair in original query is a key expression. For example in TREC7 query 351 "Falkland petroleum exploration", the bi-gram expression of "Falkland" and "exploration" in rather narrow window (less than 50 words) is a good expression. All the 14 documents which contain this pattern are relevant. What we thought is frequently appearing bi-gram pattern of query terms in top ranked documents in the pilot search may contain such expressions, and can be used to make the document level average increase. Using TSV of Okapi, we selected bi-grams in rather middle size windows, and rerank the result of pilot search.

1. Rank top 1000 documents
2. Calculate the TSV score of every bi-gram of the terms in a query in fixed window size(set 40). The top 20 documents are supposed to be relevant, and the 500-1000 ranked document are supposed to be non-relevant.
3. Sort the bi-gram by TSV score
4. Select the best N (set 5) bigram whose mutual information is over the threshold(set -5). The threshold setting of mutual information means we accepted all the best bi-gram, and we did

³We did not try Robertson's phrase weighting[7] at 1999/10/27

not consider the TSV score of the words in bi-gram.

5. Rerank the pilot search result after adding bi-gram score (tf*idf)

The effect of blind application of this techniques with this parameter setting and scoring seems to be dubious. Unexpectedly its effect on TREC4 is 10% and TREC5 5% and TREC7 3%, and we applied this techniques to some of the official TREC8 runs.

The top ranked bi-gram expressions were the combination of main theme word(title word), and the other words in most cases. For example the selected bi-gram expression in TREC8 query 406 "Parkinson's disease" were "Parkinson disease", "Parkinson symptoms", "Parkinson treat", and "Parkinson patient".

Table 3 shows the change of document level average, and average precision after QE in this case.

Cond	@10	@20 (No QE)	av-prec(QE)
With	0.50	0.350	0.452
Without	0.40	0.350	0.371

Table 3: Document level average and average precision after QE for topic 406

2.6 Query Expansion

Query Expansion was used for the ad hoc task, and small web track. The Boughanem formula[7] was used to select terms.

$$TSV = (r/R - \alpha s/S).w^{(1)} \quad (1)$$

$w^{(1)}$ is modified and more general version of Robertson/Sparck Jones weight.

The α was set 0.001, and k_4 was -0.3, k_5 was 1, and k_6 was 64. The top 20 documents in the pilot search were supposed to be relevant, and the documents ranked from 500 to 1000 were supposed to be non-relevant. The top ranked 40 words which are not included in original query, which are not included in the stopword list of SMART, whose tsv score are more than 0.003, whose df are more than 20, and whose df are less than 33000 were added to the original query.

No collection enrichment technique was used.

Twenty-One at TREC-8: using Language Technology for Information Retrieval

*Wessel Kraaij, *Renée Pohlmann and †Djoerd Hiemstra

*TNO-TPD
P.O. Box 155, 2600 AD Delft
The Netherlands
`{kraaij,pohlmann}@tpd.tno.nl`

†University of Twente, CTIT
P.O. Box 217, 7500 AE Enschede
The Netherlands
`hiemstra@cs.utwente.nl`

Abstract

This paper describes the official runs of the Twenty-One group for TREC-8. The Twenty-One group participated in the Ad-hoc, CLIR, Adaptive Filtering and SDR tracks. The main focus of our experiments is the development and evaluation of retrieval methods that are motivated by natural language processing techniques. The following new techniques are introduced in this paper. In the Ad-Hoc and CLIR tasks we experimented with automatic sense disambiguation followed by query expansion or translation. We used a combination of thesaurial and corpus information for the disambiguation process. We continued research on CLIR techniques which exploit the target corpus for an implicit disambiguation, by importing the translation probabilities into the probabilistic term-weighting framework. In filtering we extended the use of language models for document ranking with a relevance feedback algorithm for query term reweighting.

1 Introduction

Twenty-One¹ is a project funded by the EU Telematics programme, sector Information Engineering. The project subtitle is “Development of a Multimedia Information Transaction and Dissemination Tool”. Twenty-One started early 1996 and was completed in June 1999. Because the TREC ad-hoc and CLIR tasks fitted our needs to evaluate the system on the aspects of monolingual and cross-language retrieval performance, TNO-TPD and University of Twente participated under the flag of “Twenty-One” in TREC-6 and TREC-7. Because the cooperation is continued in other projects: Olive and Druid we have decided to continue our TREC participation as “Twenty-One”. For the Ad-Hoc, CLIR and SDR tasks, we used the TNO vector space engine. The engine supports several term-weighting schemes. The principal term weighting scheme we used is based on statistical language models (LM). Cf. [10] and the appendix for a more detailed description of the baseline system.

2 The Ad Hoc task

2.1 Expansion of title queries

For TREC-8 we decided to focus our experiments on title queries, because they correspond better to the average queries of current IR system users. For title queries, query expansion seems an obvious technique to improve retrieval effectiveness. We have experimented with techniques to use a lexical thesaurus for query expansion. The thesaurus is part of the VLIS lexical database of Van Dale publishers. Query expansion involves a series of steps:

1. POS tagging and lemmatization of each query word

¹Information about Twenty-one, Olive and DRUID is available at <http://dis.tpd.tno.nl/>

2. Lookup of the lemma in the VLIS lexical database. The result is a "Lexical Entity" (LE), which is a reference to a concept description in the VLIS database
3. Often the previous step results in a list of concepts, especially in the case of homonyms. In these cases some form of disambiguation is needed.
4. Expansion of the concepts with related concepts, e.g. synonyms and/or hyponyms
5. Realisation of the expansion concepts in English, using the VLIS translation relations

Unfortunately the technique did not yield convincing results on the TREC-7 topic set, so we decided to base the official TREC-8 runs on our TREC-7 system, without any of the newly developed techniques. However, these techniques were used in one of the official TREC-8 CLIR runs.

We will discuss one potential reason for the failure of query expansion techniques for title queries: One of the problems in developing a system tuned for title-only runs is the fact that the judgments for test collections are based on the full topics. The title field is a two or three word *summary* of the topic which is composed after the topic has been developed. Therefore such a title will always cover only a limited set of topic aspects. If different persons would interpret the title query, they would have different interpretations of the relevance of retrieved documents. That is because title queries are necessarily under-specified. But because the judgments have been done with the full topic description in mind (including detailed constraints when a document is - or is not - relevant) it is hard to devise a query expansion method which will improve average precision of a title run, because it is hard to predict which query constraints are described in the description and narrative sections of the topic.

2.2 Experimental setup and results

For the official runs in the ad-hoc task we eventually re-used our TREC-7 system, because the experimental query expansion systems scored significantly worse on the TREC-7 test collection. The TNO vector space engine was configured to use LM weighting using an λ_i of 0.15 and standard Porter stemming. Stop-words were removed from the documents, including words that are frequent in previous TREC topics like *relevant* and *document*. Queries were generated automatically from the full topics (title, description and narrative) using the same procedure as used for indexing.

The results of our official runs in the Ad Hoc tasks are given in table 1. We submitted 3 official runs, using either only the title or both the title and description fields. We compared a baseline run with a Pseudo Relevance Feedback (PRF) run. After an initial retrieval run, the top 200 of the weighted index terms extracted from a concatenation of the top 3 documents were added to the query with a ratio of 20 : 3, i.e. the weight of the added terms was multiplied by 3/20 before adding. These parameters were determined empirically by experimenting with the English topics of the TREC-6 CLIR track.

run-name	topic fields	mode	AVP
tno8d3	t+d	PRF	0.2921
tno8d4	t+d	base	0.2778
tno8t2	t	base	0.2423
tno8t3	t	PRF	0.2755

Table 1: Ad Hoc results

3 Cross Language Information Retrieval (CLIR)

3.1 Introduction

Like in previous years, our CLIR approach is based on query translation using bilingual dictionaries. A Twenty-One cross-lingual run consists of three steps:

1. Translate the topics in the three other languages (we used the English topics as source)
2. Perform 4 parallel runs on the sub-collections, with the translated topics
3. Merge the 4 runs into a final result file

Unlike the Ad Hoc task where we used Porter stemming, we used morphological stemming based on the Xelda tools of XRCE Grenoble for all languages². We have some indications that the fact that the stemmer only removes inflectional affixes, results in reduced effectiveness. Experiments with a derivational version are planned. For German we experimented with several strategies to deal with compounds, which were initially developed for Dutch [16]. We eventually used a non-optimal strategy (i.e. the strategy which replaces a compound by its parts) because the optimal strategy (just add compound parts as index terms) interfered with the merging strategy (retrieval status values (RSVs) are not compatible). All CLIR runs used the fuzzy expansion procedure as described in [10] to catch spelling variants of proper nouns and typos.

3.2 Translation strategies

Query translation in Twenty-One is based on the VLIS lexical database developed by Van Dale Lexicography for translations into German and French and on Systran for the translation from English into Italian. We used three different strategies for selecting translations from the VLIS database: dictionary preferred, boolean and disambiguation. The dictionary preferred and boolean strategies were also used last year, the disambiguation strategy was developed for this year's participation.

3.2.1 Dictionary preferred

In the dictionary preferred translation strategy, the selection of translations is based on the number of occurrences of a certain translation in the dictionary. Some lemmas have identical translations for different senses. If this is the case, this translation is selected. If no translation occurs more than once, the first translation is chosen by default.

3.2.2 Boolean

For the boolean strategy, translations are weighted based on the number of occurrences in the dictionary. If a translation occurs in the dictionary under three senses we assign it a weight that is three times as high. As Dutch serves as an interlingua, translation can be carried out via several Dutch pivot lemmas. This possibly generates even more occurrences of the same translation. The implicit assumption made by weighting translations is that the number of occurrences generated from the dictionary may serve as rough estimates of actual frequencies in parallel corpora. Ideally, if the domain is limited and parallel corpora on the domain are available, weights should be estimated from actual data.

3.2.3 Word sense disambiguation

This year we also experimented with a word sense disambiguation technique for cross-language retrieval. In this technique, dictionary-based word senses are disambiguated using corpus information. First, the original query is used for monolingual retrieval on the TREC ad-hoc corpus. All terms in the top N documents produced by this run are saved. Subsequently, the LEs and all lexical realisations of query terms are looked up in the VLIS database. The semantic relations defined in VLIS are used to look up synonym, hyponym and hyperonym LEs of each different sense of a query term and their lexical realisations. In this way we gather a structured group of words associated with each particular sense of a query term. These groups are further expanded using words from example sentences which are also included in the database.

The groups of words for each possible sense of a query term are subsequently compared with the terms from the monolingual retrieval run and "evidence" for each sense is computed based on the overlap between the two sets of terms. The sense for which the most evidence is found is selected. If no evidence is found at all or all senses score equally, the first sense is selected by default.

²In TREC-7 we relied on the Porter stemmer for Italian, developed at ETH

LEs	hyperonym relations	synonym relations	hyponym relations
<i>bank</i>	concern undertaking business enterprise	house institute	banker deposit mortgage loan trade
<i>bank</i>	rise elevation mound		sandbank shoal aground stuck
<i>pipe</i>	object	duct funnel nozzle tube	supply drain eustachian
<i>pipe</i>		tobacco	peace clay water hookah opium

Table 2: example word groups

Query translation is now fairly straightforward. The translations for the selected word senses are looked up in the VLIS database, if more than one translation is given for a particular sense the boolean weighting strategy is applied (cf. section 3.2.2 above).

We experimented with different values of N for the initial monolingual retrieval run, 20 turned out to be the best choice. We also experimented with re-weighting words associated with a particular word sense based on their semantic relation to the original term, e.g. assign hyponyms a higher weight than hyperonyms. This experimentation provided some evidence that hyponyms are very important for sense determination but synonyms should possibly be excluded from the sense groups.

3.3 Merging Strategies

The merging strategies used for TREC-7 were a major performance bottleneck, because the merged runs scored about 75-80% of the averaged average precision of the constituting runs. We compared different merging strategies: i) naive merge: this means simply merging the result files, assuming that the RSVs are “compatible” ii) Rank based merge: This technique was applied by IBM at TREC-7 [6]. The assumption is that $\log(R)$ where R is the rank number has a linear relationship with the probability of relevance. The method estimates the linear model on training data (e.g. previous TREC collection) e.g. by applying regression and simply replaces each RSV in a run by the estimated probability of relevance which is a function of the rank. iii) combination of evidence: just add the RSVs of method i and ii.

The LM term weighting model is founded in probability theory, but the RSV’s in the implementation in the TNO vector space engine are not equivalent to the probability of relevance. The RSVs are actually a log of the probability of relevance offset by a query dependent constant and a collection dependent constant. For the naive merge method we divided the RSVs by the query length in order to compensate for differences in query length between different language versions of a topic³. The IBM merge strategy has the implicit assumption that all topics have a similar probability function of R for all languages. It’s obvious that this assumption is not optimal, because the distribution of relevant topics over the different collections is not equal, with the extreme case that some topics only have relevant documents in 1 or 2 sub-collections. Our combined strategy simply sums the original RSV (which is the log of the probability of relevance, normalized on query length but offset by some unknown constant) with the estimated log of the probability of relevance at rank R . The method empirically scored well, probably because the IBM probability estimates help to map to a normalized RSV. There is some theoretical justification because the probability at rank R $P(D|R)$ can be used as an estimate for the a-priori probability that a document is relevant $P(D)$.

3.4 Results

Table 4 lists the results for our official runs. We discovered an error in the **tno8gr** merged run, it did contain no French documents. The tables list the results for the fixed **tno8gr** run. As a baseline we included **tno8mx**, a run which is based on a merge of 4 monolingual runs. We hoped to improve the pool with this run, in order to enable a better evaluation of the monolingual and bilingual intermediate runs. The best result is achieved by **tno8gr-fixed** the boolean run. The table also lists the results of the other merging alternatives. From our preliminary analysis we conclude that for the xlingual runs the naive score based merging strategy performs always better than the interleaved or rank based probability estimates strategy. The combination of evidence approach adds some extra improvement in most cases. The rank based merging strategy is based on precision at rank R estimates of the TREC7 **tno7mx** run. However, the TREC8 topic

³Only necessary for the merged monolingual run

run-name	description
tno8dpx	dictionary preferred translation of English query into 3 other languages; fuzzy expansion of each query term
tno8gr	probabilistically interpreted boolean query of all possible translations of the English queries into 3 other languages ; fuzzy expansion
tno8dis	disambiguation and translation of English queries into 3 other languages; fuzzy expansion of each translated term
tno8mx	reference run: merged run of four monolingual searches; fuzzy expansion of each query term

Table 3: description of CLIR runs

run-name	combination of evidence official	interleave unofficial	naive unofficial	rank based unofficial
tno8dpx	0.2523	0.2049	0.245	0.2214
tno8gr-fixed	0.2789	0.2288	0.2763	0.2102
tno8dis	0.2407	0.1905	0.2355	0.1906
tno8mx	0.3226	0.3159	0.2763	0.2625

Table 4: mean average precision of CLIR runs

set has much less relevant English documents. This is probably the reason that the success of a pure rank based merging strategy is limited.

When we look at the results of the constituting runs (table 5), the results are more consistent than in TREC-7. In TREC-7 the best performing intermediate runs were the dictionary preferred runs, and the boolean run was the best merged run. In TREC-8 the boolean strategy has the best intermediate and merged average precision.

If we compare the cross-language runs with their monolingual counterparts on a per-query basis, there are a number of queries with very poor results for all three query translation strategies. We have identified some of the factors which contributed to this effect.

- The failure to recognize and translate phrases as a unit. This is especially detrimental for the English to German runs where English phrases have to be translated into German single word compounds, e.g. "World War" → "Weltkrieg", "armed forces" → "Bundeswehr" (query 61).
- Tagging errors, e.g. "arms" (weapons) was tagged as the plural of "arm" (body part) by the Xerox tagger (query 66).
- Because most words in query titles were capitalized, we decided to convert them to lower case to prevent the tagger from tagging all title words as proper nouns. This had the effect that those title words that were actually proper nouns were not tagged correctly, e.g. the proper name "Turkey" was translated as "Truthuhn" and "dindon" (bird) in German and French respectively (query 66).

Although the results with disambiguation were somewhat disappointing, we intend to continue our experiments with word sense disambiguation in the future. One possible improvement we intend to investigate would be to use the unique Lexical Entity identifiers provided by the VLIS database instead of actual words

run-name	avg.prec. english	avg.prec. french	avg.prec. german	avg.prec. italian	average over 4	merged	relat. to avg. (%)
tno8dpx	0.3130(m)	0.3319	0.2053	0.3017	0.2880	0.2523	88
tno8gr-fixed	0.3130(m)	0.3672	0.2511	0.3017	0.3080	0.2789	91
tno8dis	0.3130(m)	0.3099	0.1806	0.3017	0.2763	0.2407	87
tno8mx(m)	0.3130(m)	0.5510(m)	0.4100(m)	0.3620(m)	0.4090	0.3226	79

Table 5: per language performance and the effect of merging on 28 topics TREC-8, (m) indicates monolingual run

as a conceptual interlingua. Our current strategy has the disadvantage that after the monolingual disambiguation process, which reduces source language query terms to unique LEs, new ambiguities are introduced in the translation process when the LEs are realized as actual words in the target language.

Not surprisingly, since it was less well tested than the other two strategies which were also used last year, we also found that the disambiguation procedure contained a few omissions which resulted in the failure to translate query terms. We found that some LEs in the VLIS database did not have lexical realisations in all languages (i.e. so-called lexical gaps). In those cases the VLIS database suggests a less optimal translation. These translations were not found by the disambiguation procedure, however.

3.5 Pool validation

Judgements for the cross-language task are probably not as complete as the judgements for the other TREC tasks [10]. In this section we try to get an indication of how much of a problem the incomplete judgements actually are. For previous TREC CLIR task runs, we evaluated each run that contributed to the pool using relevance judgements both with (standard evaluation) and without the relevant documents that the run uniquely contributed to the pool.⁴ The difference between the two evaluations will give an idea of how reliable the collections are for future work.

run name	average precision		difference		unique rel.
	unjudged	judged			
98EITdes	0.1919	0.1962	0.0043	2.2 %	45
98EITful	0.2514	0.2767	0.0253	10.1 %	159
98EITtit	0.1807	0.1841	0.0034	1.9 %	27
BKYCL7AG	0.2345	0.2406	0.0061	2.6 %	44
BKYCL7AI	0.2012	0.2184	0.0172	8.6 %	120
BKYCL7ME	0.3111	0.3391	0.0280	9.0 %	164
RaliDicAPf2e	0.1405	0.1687	0.0282	20.1 %	176
TW1E2EF	0.1425	0.1569	0.0144	10.1 %	107
ceat7f2	0.1808	0.2319	0.0511	28.3 %	293
ibmcl7al	0.2939	0.3168	0.0229	7.8 %	135
lanl982	0.0296	0.0487	0.0191	64.5 %	140
tno7ddp	0.2174	0.2382	0.0208	9.6 %	152
tno7edpx	0.2551	0.2846	0.0295	11.6 %	109
umdxeof	0.1448	0.1610	0.0162	11.2 %	140
max:			0.0511	64.5 %	293
mean:			0.0205	14.1 %	129
standard deviation:			0.0124	16.1 %	67

Table 6: TREC-7 pool validation

Table 6 shows the results of the pool validation experiment. On average, an unjudged run will have 0.02 higher average precision after judging. However, the difference may be much worse, up to 0.05 for ceat7f2. It might be possible to use information about the quality of the pool like the mean and standard deviation of the differences to define a confidence interval on the average precision of unjudged runs, but that goes beyond the scope of this paper. A complicating factor is how to handle the case where a pair of judged runs from the same group uniquely found a relevant document. The runs that show the maximum absolute difference (ceat7f2) and the maximum relative difference (lanl982) come from systems of which only one run was judged. For a research group that did not participate in TREC-7, the penalty for not being able to judge the run may therefore be higher than table 6 suggests.

Table 7 shows that the total number of judged documents is more or less the same as last year. However the average number of relevant documents per topic is lower than 100. This probably means that the quality of the pool has improved, which makes the collection more useful for per language comparisons.

⁴Thanks to Chris Buckley for proposing the pool validation experiment

collection	total docs.	judged docs.	relevant docs.	no hits in topic	judged fraction	judged docs.	relevant docs.	no hits in	judged fraction
english	242,866	8,973	956	59,63,66,75	0.0013	9,810	1,689	26,46	0.0014
french	141,637	5,751	578	76	0.0014	6,130	991	-	0.0015
german	185,099	4,098	717	60,75,76	0.0008	4,558	917	26	0.0009
italian	62,359	4,334	170	60,63,75,80	0.0024	3,062	501	26,44,51	0.0018
total	631,961	23,156	2,421	average:	0.0013	23,560	4,098	average:	0.0013

Table 7: CLIR task statistics (a) 28 topics TREC-8, (b) 28 topics TREC-7

4 Adaptive filtering

In the TREC-7 filtering task three important issues turned up [5]: 1) the initial threshold, 2) threshold adaption and 3) query reweighting. Setting the thresholds probably has the greater impact on perceived performance in terms of utility [21]. Once the threshold performs satisfactory, it is hard to improve upon the performance by query reweighting. Although we put a considerable amount of work in the threshold algorithms, the main objective of the Twenty-One participation was the development of relevance weighting algorithms for the linguistically motivated probabilistic model. Details of the probabilistic retrieval model can be found in the appendix of this paper.

4.1 Evaluation setup

For the filter track we used the experimental linguistically motivated probabilistic retrieval engine developed at the University of Twente. Initial document frequencies for term weighting were collected from the '87 to '91 editions of the Wall Street Journal (TREC CDs 1 and 2). We did not use the '92 editions because this data would not have been available in a real world application. The topics and The Financial Times documents were stemmed using the Porter stemmer and stopped using the Smart stop-list which was augmented with some domain-specific stopwords like 'document' and 'relevant'. We used title, narrative and description of the topics to build the initial profile. The controlled language fields of the Financial Times test collection were not used. We did not process the incoming documents in chunks. That is, document frequencies were updated for each incoming document; a binary decision was made directly for each incoming document; selected documents were immediately checked for relevance; thresholds and profiles were immediately updated after the relevance assessments. Unjudged documents were assumed to be not relevant. All selected documents were saved for future updating of thresholds and query profiles.

4.2 Setting the initial threshold

The linguistically motivated model ranks documents by the probability that the language model of the document generates the query (see the appendix). For ranking this is sufficient, but for binary selection of a document we need to answer the question "when is the probability high enough?". One way to answer this question is to relate the probability of sampling the query from a document to the probability that the query is the result of a random sample from the entire collection. Queries that have a high probability of being sampled from the collection (i.e. queries with common words), should receive a higher initial threshold than queries with a low probability of being sampled from the collection (i.e. queries with uncommon words). We might approximate the probability that the query T_1, T_2, \dots, T_n of length n is sampled from the collection as follows.

$$P(T_1=t_1, T_2=t_2, \dots, T_n=t_n) = \prod_{i=1}^n \frac{df(t_i)}{\sum_t df(t)} \quad (1)$$

Initially only documents that generate the query with a much higher probability than equation 1 should be selected. The initial threshold might be set to select documents with probabilities that are more than 100.000 times higher than the probability of random selection. This does not result in a very high threshold, because words that appear only once in the Wallstreet Journal receive a probability smaller than 1 in 2 million according to equation 1 and the probabilities $P(T=t|D=d)$ of a term t given a document-id d are much higher for matching terms.

After rewriting the probability measures to their corresponding vector product weighting algorithms (see the appendix), the document frequencies in the initial threshold disappear. The vector product threshold that corresponds with the decision above is $threshold = n \log(1/(1 - \lambda_i)) + c$, where $c = \log(100.000)$. This shows an interesting feature of the initial threshold. In its vector product form, the threshold is related to the relevance weights λ_i . High initial relevance weights result in a high initial threshold. Relevance weights were initialised as $\lambda_i = 0.5$ and were re-estimated after feedback.

4.3 Threshold adaption

The threshold adaption algorithm is the part of the system that uses the utility functions to optimise its performance. We simply decided to aim just below the optimum utility given the similarities of the documents that were selected by the system. Updating was done as follows.

1. recompute the similarities of all selected documents (because of changed document frequencies and changed relevance weights);
2. recompute the initial threshold (because of changed relevance weights) and add it to the selected documents like it was a non-relevant document;
3. rank the selected documents by their similarities and find the maximum utility max ;
4. the new threshold will be the similarity of the lowest ranked document that has a utility of $max - 3$ when optimising for LF1 and $max - 1$ when optimising for LF2.

As long as the system does not find any relevant document, it will increase its threshold quite fast. In general, it will never lower its threshold again, although this might happen in practice because changed document frequencies and relevance weights sometimes change the ranking of selected documents.

4.4 Relevance weighting of query terms

Initially, when no information on relevant documents is available, each query term will get the same relevance weight $\lambda_i = 0.5$. So, initially we assume that the query profile is best explained if on average half of the query terms is sampled from relevant documents and the other half is sampled from the updated Wall Street Journal data. If a relevant document is available, we might be able to explain the query profile better. Query terms that occur often in the relevant document(s) are more likely to be sampled from the relevant document. They should get a higher relevance weight. Query terms that do not occur (often) in the relevant document(s) are more likely to be sampled from the Wall Street Journal data.

Notice that we cannot simply use the proportions of relevant and non-relevant documents that contain a query term to directly estimate the new relevance weight as is done in classical probabilistic models [19]. When searching for the best relevance weights, we have to take into account the term frequencies of terms in the relevant documents. A possible approach to relevance weighting is the EM-algorithm (expectation

<div style="display: flex; justify-content: space-between;"> <div style="width: 15%;">E-step:</div> <div> $m_i = \sum_{j=1}^r \frac{\lambda_i^{(p)} \cdot P(T_i = t_i D_j = d_j)}{(1 - \lambda_i^{(p)})P(T_i = t_i) + \lambda_i^{(p)}P(T_i = t_i D_j = d_j)}$ </div> </div> <div style="display: flex; justify-content: space-between; margin-top: 10px;"> <div style="width: 15%;">M-step:</div> <div> $\lambda_i^{(p+1)} = \frac{m_i + 1.5}{r + 3}$ </div> </div>

Figure 1: relevance weighting of query terms: EM-algorithm

maximisation algorithm [4]) of figure 1. The algorithm iteratively maximises the probability of the query t_1, t_2, \dots, t_n given r relevant documents d_1, d_2, \dots, d_r . Before the iteration process starts, the relevance weights are initialised to their default values $\lambda_i^{(0)} = 0.5$, where i is the position in the query. Each iteration p estimates a new relevance weight $\lambda_i^{(p+1)}$ by first doing the E-step and then the M-step until the value of the relevance weight does not change significantly anymore. The M-step should be a maximum likelihood estimate according to its definition [4], but we used a Bayesian update because a small number of relevant documents should not radically change the initial relevance weights.

4.5 Experimental results

Six official runs were submitted: three optimised for LF1 and three optimised for LF2. For both utility functions we did the same three experiments.

1. a baseline run that only uses the initial threshold setting and threshold adaption routines;
2. the same run as 1, but with relevance weighting of query terms;
3. the same run as 1, but using a very high initial threshold.

The high initial threshold experiments were done using the TNO vector engine under slightly different conditions. These two runs use the AP Newswire data for the initial estimation of document frequencies and a somewhat different stop list. We do not think that these slightly different conditions change the big picture of our evaluation results.

run name	description	LF1	LF2	prec.	recall
uttno8lf1	optimised for LF1	-9.30	4.86	0.242	0.240
uttno8lf1f	optimised for LF1; query reweighting	-7.28	7.10	0.243	0.251
uttno8lf1p	optimised for LF1; high initial threshold	-1.20	2.46	0.216	0.105
uttno8lf2	optimised for LF2	-12.96	4.80	0.232	0.254
uttno8lf2f	optimised for LF2; query reweighting	-9.12	6.60	0.237	0.254
uttno8lf2p	optimised for LF2; high initial threshold	-5.54	1.34	0.199	0.127

Table 8: adaptive filtering, official results averaged over topics

Table 8 lists the evaluation results of the official runs using four evaluation measures: LF1, LF2, precision and recall averaged over topics. Recall and precision were calculated by assigning 0 % recall to topics with no relevant documents and assigning 0 % precision to topics with empty retrieved sets. Both baseline runs show a consistent improvement in the average utility and the average precision/recall after relevance weighting of query terms.

The high initial threshold run shows different behaviour. When optimising for LF1 (uttno8lf1p), the performance in terms of average LF1 utility improves considerably. At the same time, the performance in terms of precision and recall goes down. When optimising for LF2, a high initial threshold results in a system with lower performance than the baseline in terms of average utility, precision and recall.

4.6 Some thoughts on the evaluation

The problem with the LF1 utility is that it is plain too hard to build a system that does not perform below zero utility on average. Scoring negatively on utility means that the user would prefer to use no system at all. We found ourselves deliberately worsening our filtering system (that is lowering its precision and recall) to improve the utility score up to a point where we came pretty close to no system at all. The uttno8lf1p run did not select any document for 22 out of 50 topics.

Average utility and average precision/recall both have their disadvantages if used for the evaluation of adaptive filtering runs. In short, precision causes problems with topics for which the system selected no relevant document at all, and the problem with average utility is that it will be dominated by topics with large retrieved sets [11]. We feel that utility and precision/recall are both valuable measures for the evaluation of adaptive filtering systems. In future evaluations, situations in which both measures contradict each other, like for the LF1 experiments mentioned above, should be avoided. An obvious solution would be to aim a little bit lower. The LF2 utility function seems to be a reasonable measure for future evaluations.

5 Spoken Document Retrieval

5.1 Word recognition vs. word spotting

In TREC-7, TNO [5] investigated whether effective retrieval algorithms based on phoneme recognition and a word spotter could be built. The absence of a Language Model (which is a key component in a word based

recognizer) was found to be a serious drawback. For TREC-8, LIMSI kindly provided the word recognition transcripts. For details on the speech recognition algorithms we refer to LIMSI's paper in this volume.

5.2 Olive

TNO-TPD, University of Twente and LIMSI participate in the EU project Olive. This project is a direct descendant of the Twenty-One project. Olive uses Twenty-One retrieval technology to retrieve video fragments from a video database. In order to enable an automatic indexing step of the video material, we employ automatic speech recognition on the soundtrack of the video. The recognition transcripts contain detailed timecode information which ensures a precise coupling of the transcripts with the video. For video retrieval, a user must type a query, the query is matched against an index of noun phrases extracted from the recognition transcripts. The resulting hitlist is visualized by marking hits on a bar which represents the timeline of a video. Clicking on one of these marks will start the video at the corresponding offset through a streaming server. The video material that is used in Olive is in German and French. The speech recognition for these languages is developed at LIMSI. Because the TREC SDR task is highly relevant for Olive, we decided to cooperate with LIMSI for the TREC-8 task. LIMSI provided us with transcripts of both the TREC-7 and TREC-8 SDR data, and we tuned our retrieval on the TREC-7 SDR test collection.

5.3 Relevance Feedback

We studied pseudo relevance feedback techniques that were successfully applied by other groups in TREC-7. After some testing on TREC-7 we found that a technique introduced as "Blind Relevance Feedback" [15] performed best. The relevance feedback was applied on a larger secondary corpus: the TREC Ad Hoc corpus. Even though the corpus covers a different time-span, results with the secondary corpus were better than BRF on the SDR corpus. The following BRF parameters were used:

- select top 20 documents
- compute 60 best terms based on BRF algorithm
- add new terms down-weighted with factor 20/6

5.4 Unknown story boundaries

We reviewed the literature on story segmentation but because of time pressure we were only able to implement a baseline system for unknown story boundaries, based on fixed windows. So we did not attempt to detect story boundaries at all, we simply wanted to know how a baseline system would perform. In [18] a default section window size of 250 words was recommended, this was estimated as a 15kbyte length fragment of the transcript files, because the average number of bytes per recognized word (including timecode mark-up) is about 60. The segments have an overlap of 50 bytes to avoid missing words that occur right at a window boundary.

5.5 Experimental setup and results

We tested two term weighting algorithms (LM and BM25) in combination with two automatic query expansion techniques (PRF and BRF) on the TREC-7 SDR test collection. A combination of BM25 and Blind relevance feedback (as implemented by Cambridge University[15]) yielded the best results. For TREC8 we found that LM weighting performed consistently better than BM25 in combination with BRF (see table 9). The somewhat poorer performance of LM on TREC-7 SDR can probably be attributed to the rather small size of the TREC-7 collection, the TREC-8 results are probably much more reliable.

The results⁵ for the known story boundary conditions are good, though they could be improved. It was only after the submission deadline that we discovered that quite a few topics contain proper noun abbreviations in a format which is idiosyncratic for recognizer output, e.g. *U. S.* which would normally be

⁵We list the uninterpolated average precision over 49 topics

run-name	transcript source	term weighting	mode	AVP
tno8b-r1-limsi	manual	BM25	BRF	0.4806
tno8b-b1-limsi	NIST	BM25	BRF	0.4650
tno8b-s1-limsi	LIMSI	BM25	BRF	0.4826
tno8c-r1-limsi	manual	LM	BRF	0.5169
tno8c-b1-limsi	NIST	LM	BRF	0.4898
tno8c-s1-limsi	LIMSI	LM	BRF	0.4969

Table 9: SDR results: Known Story Boundaries

spelled as *US*. Our tokenizer will remove the abbreviation dots, and the single letters will be stopped as well. What we need is a special tokenizer which recognizes these special cases.

run-name	transcript source	mode	AVP
tno8b-b1u-limsi	NIST	BRF	0.0238
tno8b-s1u-limsi	LIMSI	BRF	0.0325

Table 10: SDR results: Unknown Story Boundaries

The unknown story boundary condition yielded very poor results. This is probably due to the fact that no effort was done to merge clusters of hits into single documents. Multiple hits in the same story were quite heavily penalized in the scoring algorithm. Further analysis is needed to check this assumption.

6 Conclusions

The probabilistic retrieval model based on statistical language models performs consistently well in all tracks. The results of the experiments with sense disambiguation are slightly disappointing, although a real evaluation is only possible when the techniques are more mature. It is a question however whether the disambiguation step can be effective because documents are indexed on terms, not on senses. We improved upon our CLIR results of last year, due to a better merging technique, unfortunately our best official xlingual run (tno8gr) suffered from a merging error. Our best xlingual run uses the corpus for implicit disambiguation and interpolates between a rank and score based merging strategy. In Adaptive Filtering we showed that LM weighting can be extended with a relevance feedback algorithm. Finally, in the SDR track we showed that a word error rate of 26.3% does not harm retrieval effectiveness in a significant way when standard retrieval techniques are used.

Acknowledgements

The work reported in this paper is funded in part by the Dutch Telematics Institute project Druid and the EU project Olive (LE 8364). We would like to thank Chris Buckley for suggesting the evaluation of the cross-language pool. Furthermore we would like to thank Rudie Ekkelenkamp and Hap Kolb of TNO-TPD for their help with the SDR and filtering tracks, and LIMSI for providing their SR transcripts to us.

Appendix: using language models for document ranking

The Twenty-One TREC-8 evaluations are based on the use of statistical language models for information retrieval [8, 9, 10]. This appendix gives an overview of the model and of its application to cross-language information retrieval and adaptive filtering. Similar models were developed and evaluated by other groups participating in TREC [3, 12, 14, 17].

A.1 An informal description of the underlying ideas

When using statistical language models for information retrieval, one builds a simple language model for each document in the collection. The term “language model” refers to statistical models similar to language models used in e.g. speech recognition. Given a query, a document is assigned the probability that the language model of that document generated the query.

The metaphor of “urn models” [13] might give more insight. Instead of drawing balls at random with replacement from an urn, we will consider the process of drawing words at random with replacement from a document. Suppose someone selects one document in the document collection; draws at random, one at a time, with replacement, ten words from this document and hands those ten words (the query terms) over to the system. The system now can make an educated guess as from which document the words came from, by calculating for each document the probability that the ten words were sampled from it and by ranking the documents accordingly. The intuition behind it is that users have a reasonable idea of which terms are likely to occur in documents of interest and will choose query terms accordingly [17]. In practice, some query terms do not occur in any of the relevant documents. This can be modeled by a slightly more complicated urn model. In this case the person who draws at random the ten words, first decides for each draw if he will draw randomly from a relevant document or randomly from the entire collection. The yes/no decision of drawing from a relevant document or not, will also be assigned a probability. This probability will be called the relevance weight of a term, because it defines the distribution of the term over relevant and non-relevant documents. For ad-hoc retrieval all non stop words in the query will be assigned the same relevance weight. For adaptive filtering, the user’s feedback will be used to re-estimate the relevance weight for each query term.

The model evaluates Boolean queries by treating the sampling process as an AND-query and allowing that each draw is specified by a disjunction of more than one term. For example, the probability of first drawing the term *information* **and** then drawing either the term *retrieval* **or** the term *filtering* from a document can be calculated by the model introduced in this paper without any additional modeling assumptions. Boolean queries were used to model more than one possible translation per query term in cross-language information retrieval.

Furthermore, the model can be extended with additional statistical processes to model differences between the vocabulary of the query and the vocabulary of the documents. Statistical translation can be added to the process of sampling terms from a document by assuming that the translation of a term does not depend on the document it was sampled from. Cross-language retrieval using e.g. English queries on a French document collection uses the sampling metaphor as follows: first an French word is sampled from the document, and then this word is translated to English with some probability that can be estimated from a parallel corpus.

A.2 Definition of the corresponding probability measures

Based on the ideas mentioned above, probability measures can be defined to rank the documents given a query. The probability that a query T_1, T_2, \dots, T_n of length n is sampled from a document with document identifier D is defined by equation 2.

$$P(T_1, T_2, \dots, T_n | D) = \prod_{i=1}^n ((1 - \lambda_i)P(T_i) + \lambda_i P(T_i | D)) \quad (2)$$

In the formula, $P(T)$ is the probability of drawing a term randomly from the collection, $P(T|D)$ is the probability of drawing a term randomly from a document and λ_i is the relevance weight of the term. If a query term is assigned a relevance weight of $\lambda_i = 1$, then the term is treated as in exact matching: the system will assign zero probability to documents in which the term does *not* occur. If a query term is assigned a relevance weight of 0, then the term is treated like a stop word: the term does not have any influence on the final ranking. In section A.4 it is shown that this probability measure can be rewritten to a tfxidf term weighting algorithm. A similar probability function was used by Miller, Leek and Schwartz [12]. They showed that it can be interpreted as a two-state hidden Markov model in which λ and $(1 - \lambda)$ define the state transition probabilities and $P(T)$ and $P(T|D)$ define the emission probabilities.

The evaluation of Boolean queries for cross-language retrieval is straightforward. For each draw, different terms are mutually exclusive. That is, if one term is drawn from a document, the probability of drawing e.g. both the term *information* and the term *retrieval* is 0. Following the axioms of probability theory (see e.g. Mood [13]) the probability of a disjunction of terms in one draw is the sum of the probabilities of drawing the single terms. Disjunction of m possible translations T_{ij} ($1 \leq j \leq m$) of the source language query term on position i is defined as follows.

$$P(T_{i1} \cup T_{i2} \cup \dots \cup T_{im} | D) = \sum_{j=1}^m ((1 - \lambda_i)P(T_{ij}) + \lambda_i P(T_{ij} | D)) \quad (3)$$

Following this line of reasoning, AND queries are interpreted similar as unstructured queries defined by equation 2. Or, to put it differently, unstructured queries are implicitly assumed to be AND queries. If a relevance weight of $\lambda_i = 1$ is assigned to each query term, then the system will behave like the traditional Boolean model of IR. Statistical translation is added to these probability measures by assuming that the translation of a term does not depend on the document it was drawn from [9]. If N_1, N_2, \dots, N_n is a English query of length n and a English term on position i has m_i possible French translations T_{ij} ($1 \leq j \leq m_i$), then the ranking as structured queries would be done according to equation 4

$$P(N_1, N_2, \dots, N_n | D) = \prod_{i=1}^n \sum_{j=1}^{m_i} P(N_i | T_{ij}) ((1 - \lambda_i)P(T_{ij}) + \lambda_i P(T_{ij} | D)) \quad (4)$$

The translation probabilities $P(N_i | T_{ij})$ can be estimated from parallel corpora, or alternatively by using occurrences in a machine readable dictionary. A very similar model that also combines document ranking and statistical translation was introduced by Berger and Lafferty [2, 3]. Their model differs from equation 4 only by a different smoothing method, using global information on N_i instead of global information on each T_{ij} .

A.3 Parameter estimation

In information retrieval it is good practice to use the term frequency and document frequency as the main components of term weighting algorithms. Our probabilistic model does not make an exception. The term frequency $tf(t, d)$ is defined by the number of times the term t occurs in the document d . The document frequency $df(t)$ is defined by the number of documents in which the term t occurs. Estimation of $P(T)$ and $P(T|D)$ in equation 2, 3 and 4 was done as follows:

$$P(T_i = t_i) = \frac{df(t_i)}{\sum_t df(t)} \quad (5)$$

$$P(T_i = t_i | D = d) = \frac{tf(t_i, d)}{\sum_t tf(t, d)} \quad (6)$$

The value of the relevance weights λ_i might change for different applications. High relevance weights result in tf×idf rankings that obey the conditions of coordination level ranking [10], that is, documents containing n query terms are always ranked above documents containing $n - 1$ query terms. High relevance weights are a good choice for applications that aim at high precision or applications in which very short queries are used, like web search engines. Documents that are judged as relevant by the user can be used to re-estimate the relevance weights. An algorithm for relevance weighting was developed for the adaptive filtering task (see section 4).

A.4 Some notes on the implementation

Similar to traditional probabilistic models of information retrieval [19] probability measures for ranking documents can be rewritten into a format that is easy to implement. A presence weighting scheme (as opposed to a presence/absence weighting scheme) assigns a zero weight to terms that are not present in a document. Presence weighting schemes can be implemented using the vector product formula. This section

presents the resulting algorithms. Rewriting equation 2 results in the formula displayed in figure 2 [10]. It can be interpreted as a tf×idf weighting algorithm with document length normalisation as defined by Salton and Buckley [20].

vector product formula:	$\text{score}(d, q) = \sum_{k \in \text{matching terms}} w_{qk} \cdot w_{dk}$
query term weight:	$w_{qk} = \text{tf}(k, q)$
document term weight:	$w_{dk} = \log\left(1 + \frac{\text{tf}(k, d) \sum_t \text{df}(t)}{\text{df}(k) \sum_t \text{tf}(t, d)} \cdot \frac{\lambda_k}{1 - \lambda_k}\right)$

Figure 2: tf×idf term weighting algorithm

If a structured query is used, the disjunction of possible translations as defined by equation 3 should be calculated first. As addition is associative and commutative, we do not have to calculate each linear interpolation of equation 3 separately before summing them. Instead, the document frequencies and the term frequencies of the disjuncts respectively, can be added beforehand. The added frequencies can be used to replace $\text{df}(k)$ and $\text{tf}(k, d)$ in the weighting formula of table 2. A similar ranking algorithm for Boolean queries was introduced earlier by Harman [7] for on-line stemming. Harman did not present her algorithm as an extension of Boolean searching, but instead called it 'grouping'. Instead of adding the document frequencies, the TNO vector engine calculates the actual document frequencies of the disjuncts, by merging their postings at run time. A similar approach for cross-language information retrieval was adopted by Ballesteros and Croft [1] by using a 'synonym operator' on possible translations.

If translation probabilities are used following equation 4, the adding of respectively the document frequencies and the term frequencies of the disjuncts should be done as a weighted sum with the translation probabilities as weights.

References

- [1] L. Ballesteros and W.B. Croft. Resolving ambiguity for cross-language retrieval. In *Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, pages 64–71, 1998.
- [2] A. Berger and J. Lafferty. The Weaver system for document retrieval. In *Proceedings of the 22nd ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 222–229, 1999.
- [3] A. Berger and J. Lafferty. The Weaver system for document retrieval. In *Proceedings of the seventh Text Retrieval Conference TREC-8*, 2000. elsewhere in this volume.
- [4] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em-algorithm plus discussions on the paper. *Journal of the Royal Statistical Society*, 39(B):1–38, 1977.
- [5] R. Ekkelenkamp, W. Kraaij, and D. van Leeuwen. TNO TREC-7 site report: SDR and Filtering. In *Proceedings of the seventh Text Retrieval Conference TREC-7*, pages 519–526, 1999. NIST Special Publication 500-242.
- [6] M. Franz, J.s. McCarley, and S. Roukos. Ad hoc and multilingual information retrieval at IBM. In *Proceedings of the seventh Text Retrieval Conference TREC-7*, pages 157–168, 1999. NIST Special Publication 500-242.
- [7] D. Harman. How effective is suffixing? *Journal of the American Society for Information Science*, 42(1):7–15, 1991.

- [8] D. Hiemstra. A probabilistic justification for using tf.idf term weighting in information retrieval. *International Journal on Digital Libraries*, 1999. (to appear).
- [9] D. Hiemstra and F.M.G. de Jong. Disambiguation strategies for cross-language information retrieval. In *Proceedings of the third European Conference on Research and Advanced Technology for Digital Libraries*, pages 274–293, 1999.
- [10] D. Hiemstra and W. Kraaij. Twenty-One at TREC-7: Ad-hoc and cross-language track. In *Proceedings of the seventh Text Retrieval Conference TREC-7*, pages 227–238, 1999. NIST Special Publication 500-242.
- [11] D. Hull. The TREC-7 filter track: Description and analysis. In *Proceedings of the seventh Text Retrieval Conference TREC-7*, pages 33–36, 1999. NIST Special Publication 500-242.
- [12] D.R.H. Miller, T. Leek, , and R.M. Schwartz. BBN at TREC-7: using hidden markov models for information retrieval. In *Proceedings of the seventh Text Retrieval Conference, TREC-7*, 1999. NIST Special Publication 500-242.
- [13] Mood and F.A. Graybill, editors. *Introduction to the Theory of Statistics, Second edition*. McGraw-Hill, 1963.
- [14] K. Ng. A maximum likelihood ratio information retrieval model. In *Proceedings of the seventh Text Retrieval Conference TREC-8*, 2000. elsewhere in this volume.
- [15] Karen Sparck Jones Pierre Jourlin, Sue E. Johnson and Philip C. Woodland. General query expansion techniques for spoken document retrieval. In *Proceedings of the ESCA ETRW Workshop: Accessing Information in Spoken Audio*, 1999.
- [16] Renée Pohlmann and Wessel Kraaij. The effect of syntactic phrase indexing on retrieval performance for Dutch texts. In L. Devroye and C. Chrismont, editors, *Proceedings of RIAO'97*, pages 176–187, 1997.
- [17] J.M. Ponte and W.B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, 1998.
- [18] Jeffrey C. Reynar. *Topic Segmentation: Algorithms and Applications*. PhD thesis, University of Pennsylvania, 1998.
- [19] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.
- [20] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. In *Information Processing & Management*, volume 24, pages 513–523, 1988.
- [21] C. Zhai, P. Jansen, E. Stoica, N. Grot, and D.A. Evans. Threshold calibration in CLARIT adaptive filtering. In *Proceedings of the seventh Text Retrieval Conference TREC-7*, pages 149–156, 1999. NIST Special Publication 500-242.

English-German Cross-Language Retrieval for the GIRT Collection – Exploiting a Multilingual Thesaurus

Fredric C. Gey and Hailing Jiang
UC Data Archive & Technical Assistance (UC DATA)
University of California, Berkeley
gey@ucdata.berkeley.edu, hjiang1@sims.berkeley.edu

Abstract

For TREC-8, the Berkeley experiments concentrated on the special GIRT collection. We utilized the GIRT thesaurus in multiple ways in working on English-German Cross-Language IR. Since the GIRT collection is truly multilingual (documents contain both German and English text), one would expect multilingual queries to achieve the best performance. This proved not to be the case.

1 Introduction

Successful cross-language information retrieval (CLIR) combines linguistic techniques (phrase discovery, machine translation, bilingual dictionary lookup) with robust monolingual information retrieval. The Berkeley group has been using the technique of logistic regression from the beginning of the TREC series of conferences. In TREC-2 [2] we derived a statistical formula for predicting probability of relevance based upon statistical clues contained with documents, queries and collections as a whole. This formula was used for document retrieval in Chinese[3] and Spanish in TREC-4 through TREC-6. We utilized the identical formula for English queries against German documents in the cross-language track for TREC-6. In TREC-7 the formula was also used for cross-language runs over multiple European languages. During the past year the formula has proven well-suited for Japanese and Japanese-English cross-language information retrieval[4], even when only trained on English document collections. Our participation in the NTCIR Workshop in Tokyo (<http://www.rd.nacsis.ac.jp/~ntcadm/workshop/work-en.html>) led to different techniques for cross-language retrieval, ones which utilized the power of human indexing of documents to improve retrieval via bi-lingual lexicon development and a form of text categorization which associated terms in documents with humanly assigned index terms[1].

2 The GIRT Collection

GIRT collection contains German documents (some have English sections inside) from the field of social science. It has some special features that make it ideal to try out different ideas. Among them are:

1. Each GIRT document was manually assigned controlled terms which are from the Social Science Thesaurus. Figure 1 shows a sample GIRT document.

On average there are about 10 terms given to a document. This offers an opportunity to explore how to utilize controlled vocabulary to enhance retrieval effectiveness.


```

<DOC>
<DOCNO>
GIRT950410185
</DOCNO>
<TITLE>
Ausländerinnen in der beruflichen qualifizierung - eine Handreichung
</TITLE>
<TITLE-ENG>
Female aliens in occupational qualification : a guide
</TITLE-ENG>
<AUTHOR>
Djafari, Nader; Brüning, Gerhild
</AUTHOR>
<DOCTYPE>
Sonstiges
</DOCTYPE>
<YEAR>
1994
</YEAR>
<PLACE>
Frankfurt am Main
</PLACE>
<CY>
DEU
</CY>
<ISBN>
3-88513-492-6
</ISBN>
<LANGUAGE>
DE
</LANGUAGE>
<CONTROLLED-TERM>
Ausländer; Frau; Beruf; Qualifikation; Bildungschance; Ausbildungssituation; Bundesrepublik Deutschland
</CONTROLLED-TERM>
<CLASSIFICATION>
Arbeitsmarkt- und Berufsforschung *
</CLASSIFICATION>
<METHOD>
Dokumentation
</METHOD>
<FREE-TERM>
GIRT
</FREE-TERM>
<CORPORATE-SOURCE>
Deutscher Volkshochschul-Verband e.V. Pädagogische Arbeitsstelle
</CORPORATE-SOURCE>
<TEXT>
"Die Handreichung gibt durch Hintergrundinformationen und Erfahrungsberichte Anregungen für Personen,
die in der beruflichen Erwachsenenbildung tätig sind, damit die Qualifizierungsmöglichkeiten auch für Frauen
aus nicht-deutschen Kulturbereichen verstärkt geöffnet werden. Die dargestellten Praxisbereiche basieren auf
Modellversuchen und einzelnen innovativen Projekten. Sie sind punktuelle Impulsgeber in einer
Weiterbildungslandschaft, die für die Zielgruppe \Ausländerinnen\ unzureichend ausgestattet ist."
(Autorenreferat, IAB-Doku)
</TEXT>
</DOC>

```

Figure 1: Sample GIRT Document, TREC-8 CLIR.

2. There are a total of 37637 documents in GIRT collection. Among them, about 27458 (73%) have both German and English titles. About 2714 (7%) have corresponding German and English text sections (abstracts). This feature would make it possible to apply some multilingual corpus techniques to create a specialized bilingual dictionary.

3 Approaches to GIRT Retrieval

3.1 Experimental setup

The GIRT collection was used in our experiments. Both German and English sections in a document were indexed. For the German sections, no stemmer was used. Single words were indexed except for Controlled-term section, in which the whole terms (phrases usually) were indexed in addition to the single word components. For the English sections of a document, the SMART stemmer was used. Single words were indexed, and no phrases were used.

3.2 Query translation

In CLIR, essentially either queries or documents or both need to be translated from one language to another. Query translation is usually selected for practical reasons of efficiency. In our GIRT experiments, we tried the following approach to translate the English query to German:

Thesaurus lookup. The social science Thesaurus is a German-English bilingual thesaurus. Each German item in this thesaurus has a corresponding English translation. We took the following steps to translate the English query to German by looking up the thesaurus:

- a. Create an English-German transfer dictionary from the Social Science Thesaurus. This transfer dictionary contains English items and their corresponding German translations. This "vocabulary discovery" approach was taken by Eichmann, Ruiz and Srinivasan for medical information cross-language retrieval using the UMLS Metathesaurus[5].

- b. Use the part-of-speech tagger LT-POS developed by University of Edinburgh (<http://www.ltg.ed.ac.uk/software/pos/index.html>) to tag the English query and identify noun phrases in the English query. One problem with thesaurus lookup is how to match the phrasal items in a thesaurus. We took a simple approach to deal with this problem: use POS tagger to identify noun phrases.

- c. Look up the single words and noun phrases in the English query in the English-German transfer dictionary. In our experiments, we found that in some cases mismatch was caused by the different formats of words used in the query and the dictionary. For example, "women" is not found in the dictionary, but "woman" is. "anti-semitism" is not in the dictionary, but "antisemitism" is. So we adopted some rules when looking up the dictionary, such as, If a word or phrase is not found in the dictionary, look up its base form. The base form of a word is obtained using WordNet. If a word with '-' inside is not found in the dictionary, replace the '-' with a space or remove the '-'.

In our experiments, over 60% of query words or phrases were found in the transfer dictionary. Those which were not found are mostly very general terms and may not have affected the retrieval result.

3.3 Query expansion

We tested two approaches to expand the translated query.

1. Use of thesaurus terms to expand queries. We tried a KNN-similarity method [6, 8] to assign German thesaurus terms to each English query and add the thesaurus terms to the the German query translated using the thesaurus lookup. First, we run the English query against the documents which have English titles and/or abstracts (using Berkeley TREC2 formula), then extract the thesaurus terms assigned to the top 30 retrieved documents, and rank them by the number of documents to which they are assigned. The top thesaurus terms that occur in at least 5 documents are chosen and added to the translated German query.

2. Use of the hierarchical relationship in the thesaurus to expand the query. For each German thesaurus term in the translated query, we add its narrow terms (NT) to expand the query.

4 GIRT Experiments - official runs

We submitted 5 official runs. The only difference between these runs is how the query was constructed to run against the collection:

BKCLGR01: English query translated to German using thesaurus lookup.

BKCLGR02: English query translated to German using thesaurus lookup and expanded by narrow terms in the thesaurus.

BKCLGR03: English query translated to German using thesaurus lookup and expanded by German thesaurus terms.

BKCLGR04: English query + German query translated using thesaurus lookup.

BKCLGR05: English query + German query translated using thesaurus lookup and expanded by German thesaurus terms.

The results of our five official runs are presented in Table 1.

Run ID	BKCLGR01	BKCLGR02	BKCLGR03	BKCLGR04	BKCLGR05
Retrieved	28000	28000	28000	28000	28000
Relevant	1294	1294	1294	1294	1294
Rel-ret	907	733	936	890	921
Precision					
at 0.00	0.6564	0.4124	0.6671	0.7111	0.7036
at 0.10	0.5326	0.3329	0.5402	0.5492	0.5325
at 0.20	0.4485	0.2826	0.4721	0.3388	0.3689
at 0.30	0.3858	0.2568	0.4087	0.2774	0.2854
at 0.40	0.2924	0.1971	0.3131	0.2068	0.2327
at 0.50	0.2640	0.1746	0.2864	0.1550	0.1768
at 0.60	0.2107	0.1175	0.2163	0.1103	0.1435
at 0.70	0.1674	0.0935	0.1726	0.0848	0.1159
at 0.80	0.1242	0.0774	0.1241	0.0572	0.0805
at 0.90	0.0412	0.0131	0.0314	0.0178	0.0319
at 1.00	0.0208	0.0018	0.0008	0.0008	0.0004
Avg prec.	0.2707	0.1667	0.2832	0.2049	0.2232

Table 1: Results of five official GIRT runs.

These results show that adding the original English terms to the queries reduced the overall precision of the results while modestly increasing the precision for the first few documents.

5 Other GIRT Experiments - unofficial runs

We continued to make other runs on the GIRT collection, exploring a variety of approaches and also creating some baseline monolingual runs against which to measure our cross-language techniques. In TREC-7 we made use of commercial machine translation software to do all runs and achieved better results than bilingual dictionary lookup. In addition to the 5 official runs, we also did these experiments:

1. SYSTRAN Machine Translation System [7]
2. SYSTRAN translation expanded by thesaurus terms
3. German monolingual
4. German monolingual expanded by thesaurus terms
5. English query directly run against the collection (without translation)

The results for these five experimental runs are shown in Table 2.

These results, when compared with the official runs, show that the vocabulary provided by the GIRT Thesaurus supplied considerable improvement over general machine translation unaugmented by a specialized dictionary. Most surprisingly, we found that the general purpose SYSTRAN translation did not perform as well as the untranslated English query.

Run ID	SYSTRAN	SYSTRAN W/Expansion	German Monolingual	Monolingual W/Expansion	English Only
Retrieved	28000	28000	28000	28000	28000
Relevant	1294	1294	1294	1294	1294
Rel-ret	644	818	838	918	545
Precision					
at 0.00	0.4057	0.5302	0.8463	0.7966	0.6802
at 0.10	0.2407	0.3244	0.6554	0.6067	0.4234
at 0.20	0.2078	0.2885	0.5601	0.4856	0.1509
at 0.30	0.1531	0.2504	0.4415	0.4327	0.1238
at 0.40	0.1033	0.1878	0.3037	0.3637	0.1141
at 0.50	0.0849	0.1642	0.2398	0.3058	0.0690
at 0.60	0.0635	0.1144	0.1480	0.2159	0.0289
at 0.70	0.0454	0.0836	0.0842	0.1409	0.0216
at 0.80	0.0228	0.0424	0.0558	0.0741	0.0078
at 0.90	0.0061	0.0159	0.0228	0.0237	0.0000
at 1.00	0.0000	0.0000	0.0122	0.0084	0.0000
Avg. prec.	0.1063	0.1654	0.2860	0.2960	0.1211

Table 2: Results of Other GIRT runs

It is interesting to note that while overall precision of the German monolingual run with query expansion (0.2960) is better than that of our best official run BKCLGR03, the official run finds more relevant documents (936 versus 918) in the top 1000 than the monolingual run.

6 Conclusions and Acknowledgments

There are many document collections available in the growing digital library world which have been humanly indexed from a controlled vocabulary. Retrieval techniques which exploit this indexing to improve retrieval are in their infancy. The TREC-8 GIRT collection provides an interesting example of how such indexing may be utilized for cross-language information retrieval if indexing is done from a multi-lingual thesaurus. We conclude that exploiting the special vocabulary features of the thesaurus can more than double retrieval precision over general purpose machine translation. We also find that using a multilingual query to search multilingual documents may not achieve the best possible performance. Furthermore we find that query expansion using narrower terms from a thesaurus may degrade performance. This is probably because the extra terms seem to add noise documents to the retrieved set. It remains to be seen whether the inherent structure of the thesaurus can be successfully utilized to improve retrieval performance.

For future research it would be useful to take the GIRT German/English titles and align them to create a bilingual lexicon and see how that would perform against the multilingual thesaurus approach. We are also working on applying promising text categorization techniques, which have worked in Japanese-English CLIR, for query expansion [1].

This research was supported by the Information and Data Management Program of the National Science Foundation under grant IRI-9630765 from the Information and Data Management program of the Computer and Information Science and Engineering Directorate. Partial support was also provided by DARPA (Department of Defense Advanced Research Projects Agency) under research contract N66001-97-C-8541, AO-F477.

We thank Aitao Chen for much helpful advice and some programming support.

References

- [1] F Gey A Chen and H Jiang. Applying Text Categorization to Vocabulary Enhancement for Japanese-English Cross-Language Information Retrieval. In S. Annandiou, editor, *The Seventh Machine Translation Summit, Workshop on MT for Cross-language Information Retrieval, Singapore*, pages 35–40, September 1999.
- [2] W Cooper A Chen and F Gey. Full Text Retrieval based on Probabilistic Equations with Coefficients fitted by Logistic Regression. In D. K. Harman, editor, *The Second Text REtrieval Conference (TREC-2)*, pages 57–66, March 1994.
- [3] A Chen J He L Xu F Gey and J Meggs. Chinese Text Retrieval Without Using a Dictionary. In A. Desai Narasimhalu Nicholas J. Belkin and Peter Willett, editors, *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Philadelphia*, pages 42–49, 1997.
- [4] A Chen F Gey K Kishida H Jiang and Q Liang. Comparing Multiple Methods for Japanese and Japanese-English Text Retrieval. In N. Kando, editor, *The First NTCIR Workshop on Japanese Text Retrieval and Term Recognition, Tokyo Japan*, pages 49–58, September 1999.
- [5] D Eichmann M Ruiz and P Srinivasan. Cross-Language Information Retrieval with the UMLS Metathesaurus. In W B Croft A Moffat C J van Rijsbergen R Wilkinson and J Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia*, pages 72–80, August 1998.
- [6] S Dumais J Platt D Heckerman M Sahamii. Inductive learning algorithms and representations for text categorization. In G Gardarin J French N Pissinou K Makki and L Bouganim, editors, *Proceedings of CIKM98: The Seventh International Conference on Information and Knowledge Management, Nov 3-7, 1998, Bethesda MD*, pages 148–155, November 1998.
- [7] SYSTRAN: <http://babelfish.altavista.digital.com/>.
- [8] Y. Yang and X. Lin. A Re-examination of Text Categorization Methods. In Fredric Gey Marti Hearst and Richard Tong, editors, *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Berkeley*, pages 42–49, 1999.

ACSys TREC-8 Experiments

David Hawking*

CSIRO Mathematics and Information Sciences,
Canberra, Australia

Peter Bailey and Nick Craswell

Department of Computer Science, ANU
Canberra, Australia

David.Hawking@cmis.csiro.au, {peterb,nick}@cs.anu.edu.au

October 12, 1999

Abstract

Experiments relating to TREC-8 Ad Hoc, Web Track (Large and Small) and Query Track tasks are described and results reported. Due to time constraints, only minimal effort was put into Ad Hoc and Query Track participation. In the Web Track, Google-style PageRanks were calculated for all 18.5 million pages in the VLC2 collection and for the 0.25 million pages in the WT2g collection. Various combinations of content score and PageRank produced no benefit for TREC style ad hoc retrieval. A major goal in the Web Track was to make engineering improvements to permit indexing of the 100 gigabyte collection and subsequent query processing using a single PC. A secondary goal was to achieve last year's performance (obtained with eight DEC Alphas) with less recourse to effectiveness-harming optimisations. The main goal was achieved and indexing times are comparable to last year's. However, effectiveness results were worse relative to last year and query processing times were approximately double.

1 Introduction

The work reported here comprises a number of text retrieval experiments conducted within the framework of TREC-8 and addressing questions of interest in the following research areas: Practical information retrieval; Exploitation of link information.

ACSys completed Automatic Adhoc, Query Track, Large Web and Small Web tasks.

1.1 Basic Relevance Scoring Method

As in TREC-6 and TREC-7 [Hawking et al. 1997], the basic relevance scoring method used in official ACSys adhoc runs was the Cornell variant of the Okapi BM25 weighting function [Singhal et al. 1995; Robertson et al. 1994]

$$w_t = q_t \times tf_d \times \frac{\log\left(\frac{N-n+0.5}{n+0.5}\right)}{2 \times (0.25 + 0.75 \times \frac{dl}{avdl}) + tf_d} \quad (1)$$

*The authors wish to acknowledge that this work was carried out within the Cooperative Research Centre for Advanced Computational Systems established under the Australian Government's Cooperative Research Centres Program.

where w_t is the relevance weight assigned to a document due to query term t , q_t is the weight attached to the term by the query, tf_d is the number of times t occurs in the document, N is the total number of documents, n is the number of documents containing at least one occurrence of t , dl is the length of the document and $avdl$ is the average document length (measured either in bytes or in indexable words).

1.2 Hardware and Software Employed

Two different versions of PADRE software, known as PADRE98 and PADRE99 were used in experiments reported here. An Intel PC with 1 GB of RAM and the Linux operating system was used throughout.

1.3 Query Expansion

A set of synonyms derived by manual inspection of 150 past topics was used in some of the runs with PADRE99.

Relevance feedback, as described in [Hawking et al. 1997] was used in the PADRE98 Ad Hoc runs. A more efficient implementation of the same model was used in some of the PADRE99 runs.

1.4 Retrieval in a Production Environment

Many users of Web search engines pose short queries in which small lexical differences in documents and queries are far more significant than they are with long TREC topics. Some of these tiny but potentially vital lexical signals may be eliminated by operations which normally contribute to successful TREC ad hoc participation: stemming, stopword elimination and case folding. Examples of queries in which such differences may be important include: "the Pope", "to be or not to be", "new Apples", and Hawking.

We hypothesise that stemming, stopword elimination and case folding operations should be applied (when appropriate) during query processing rather than during indexing. The PADRE99 runs reported here relate to indexes which, although case folded, are unstemmed and include all words, numbers and letter-digit combinations.

Another feature of everyday Web search engine usage is that users who enter short queries tend to be dissatisfied when results at the top of the ranking do not contain all of the query terms. For queries of less than five words, the PADRE99 software always presents documents containing more of the query terms ahead of those with less.

1.5 Statistical Testing of Differences Between Runs

Throughout this paper, wherever comparisons are made between pairs of runs, apparent differences between means have been tested for statistical significance using two-tailed t -tests with $\alpha = 0.05$.

2 Automatic AdHoc Runs

In the following the codes T, D and N are used to indicate use of Title, Description and Narrative fields of the Ad Hoc topic statements. TDN implies use of all three fields.

Figure 1 reports results for four PADRE98 runs made using the same methods used by ACSys in TREC-7. Almost identical source code was used but two changes were made to overcome problems encountered in TREC-7: The maximum length of a word recorded in the index was increased from 12 to 16 characters and the stop word list was considerably shortened.

Comparing feedback and no-feedback versions of the PADRE98 TDN runs, differences in average precision, precision at 20 documents retrieved, and recall were all statistically significant (+11%, +6%, +7% respectively).

Table 1: Performance of ANU/ACSys Automatic Adhoc runs using PADRE98 software and TREC-7 methods. Stopwords were not included in the index. (The stopwords list included 88 stems.) Index words were stemmed. Query term weights were assigned on the basis of frequency within previous queries. Pseudo-phrases (ie. word pairs) were automatically generated for TD and TDN and concept scoring ($k = 1$) for the T runs. Relevance feedback used the top 20 new terms derived from hotspots (defined as the text within 500 characters of a query word occurrence) in the top 20 documents found by the original query. The stopwords list included 88 stems.

Run-id	Topic Fields	Ave Prec	P@20	Recall	Notes
acsys8alo	TDN	.2935	.4400	.7504	No relevance feedback Unofficial run, 3 Aug 99
acsys8alo2	TDN	.2637	.4160	.7041	
acsys8amo	TD	.2792	.4260	.7524	
acsys8aso	T	.2740	.4280	.6978	

Table 2: Performance of ANU/ACSys Automatic Adhoc runs using PADRE99 software. Stopwords were not excluded from the index and index words were unstemmed. Query term weights were just the occurrence frequency within the current query. Relevance feedback used the top 30 new terms derived from hotspots (defined as the text within 100 indexable words of a query word occurrence) in the top 20 documents found by the original query. The best feedback term received 0.75 of the query-term weight assigned to a query term which occurred only once in the initial query.

Run-id	Topic Fields	Ave Prec	P@20	Recall	Notes
acsys8aln2	TDN	.2560	.4060	.6955	Corresponds to acsys8alo2
acsys8amn	TD	.2353	.3790	.6187	Synonym expansion and relevance feedback
acsys8asn	T	.2309	.3790	.5931	Synonym expansion and relevance feedback

Comparing the three feedback runs, the only significant differences on any of the measures between T, TD and TDN runs are the recall differences between T and each of the longer-topic runs (+8% in both cases).

Queries used in the acsys8aln2 run were the ones generated for acsys8alo2, but translated into the PADRE99 query language. A stem-matching operator was appended to each literal in the acsys8alo2 queries to ensure that the effect of stemming was achieved despite the fact that PADRE99 indexes were unstemmed.

There was no statistical difference on any of the three effectiveness measures between the two runs. However, the acsys8aln2 queries required an average of 12.98 sec. to run compared with 4.29 sec. per query for acsys8alo2, on the same hardware.

Query-time stemming is slower because a potentially large chunk of the term dictionary must be scanned for terms which stem to the target and multiple postings lists accessed instead of just one. The problem is worse than might be imagined because there are often a surprising number of "words" (including misspellings) which share a common stem, according to rule-based stemming (Porter method).

2.1 Follow-up Runs

Further runs were conducted post-hoc to determine the effectiveness of relevance feedback and synonym expansion as used in the PADRE99 runs. Results are tabulated in Figure 1 but have not yet been statistically analysed. On the surface, it appears that synonym expansion (as implemented) had negligible effect and that relevance feedback (as implemented) was clearly beneficial. Increasing the amount of topic text appears to improve performance but the benefit of relevance feedback appears to diminish as the length of the topic text increases, particularly on the precision dimension.

3 Query Track

A set of short queries was generated manually by David Hawking for contribution to the track pool. Hastily constructed `perl` scripts were used to translate the sets of queries into a form suitable for processing by PADRE99. No stemming was applied, no stopwords were eliminated and no particular smarts (forgive the pun) were applied during processing.

4 Small Web Task

The major questions addressed by this track were:

1. Are the best methods for retrieval over the ad hoc data also the best for the WT2g collection?
2. Can link information be used to enhance retrieval?

The ACSys contribution to answering the first question was to run the `acsys8mn` query set over the WT2g data using the identical processing parameters as had been used in the Ad Hoc track. The resulting run was called `acsys8wm`. An answer to the question can only arise from a study of the collection of runs.

ACSys contributed to the second question by combining PageRank [Brin and Page 1998] scores with the content scores generated by the `acsys8wm` run. For each topic, each document's content score was normalised so that the highest-scoring document scored 1.0. PageRank scores (which are topic independent) were scaled relative to the highest PageRank score. Normalised content and PageRank scores were treated as orthogonal axes and each document was represented as a point in 2-space. The document's final score was taken as the vector distance of that point from the origin.

Cursory inspection indicated that differences between the baseline rankings and rankings obtained by this means were small. Accordingly, in some experiments reported below, the normalised PageRank scores were multiplied by a factor of 10.0 in order to both create an effect large enough to measure and to increase the chance that pages with high PageRank scores would be judged.

4.1 How were PageRanks Computed?

PageRank is a measure of "link popularity" within a set of hyper-text documents. One way to understand the concept of link popularity is to assume a "random surfer" is walking the graph of Web pages in question, following hyper-links at random. Specifically the surfer has the following behaviour:

1. The surfer has some bookmarks, a subset of the available pages. (In the experiments reported here the complete set of available pages constituted the bookmarks except in one case where only the root pages of each of the 953 servers was bookmarked.)
2. The surfer picks a random page from the bookmarks and visits it.
3. If the visited page has no links to other pages, go to step 2

Results tabulated on Wed Oct 6 15:55:18 1999 on peace.anu.edu.au

Average Precision

	plain	-rf	-syn	rf/syn	average
short	0.1976	0.2317	0.1951	0.2310	0.2139
medium	0.2124	0.2324	0.2134	0.2341	0.2231
long	0.2209	0.2422	0.2241	0.2404	0.2319
average	0.2103	0.2354	0.2109	0.2352	

Precision @ 20

	plain	-rf	-syn	rf/syn	average
short	0.3520	0.3900	0.3490	0.3800	0.3678
medium	0.3800	0.3930	0.3790	0.3780	0.3825
long	0.3910	0.4160	0.3790	0.4000	0.3965
average	0.3743	0.3997	0.369	0.386	

Topic-by-topic recall

	plain	-rf	-syn	rf/syn	average
short	0.5361	0.5977	0.5288	0.5931	0.5639
medium	0.5691	0.6171	0.5686	0.6132	0.592
long	0.5903	0.6311	0.5852	0.6232	0.6075
average	0.5652	0.6153	0.5609	0.6098	

Figure 1: Follow-up runs with PADRE99 on the Automatic Ad Hoc task, exploring the effectiveness of synonym expansion and relevance feedback for each topic length.

4. Otherwise, pick a link at random, visit it and go to step 3.

The PageRank of a page is the probability that the surfer will be visiting that page at any point in time.

In a simple system with two pages that link to each other, the probability that a surfer will be at one page is 0.5. In a more complex system, the probability that the surfer will be at a page is roughly proportional to the in-degree of that page. If a page has no incoming links and is not on the bookmarks, the probability of a visit (and hence the PageRank) is zero. If every page has two links, but one link from every page points to page A, page A will have a very high PageRank, and the page pointed to by A will also inherit a high PageRank.

PageRanks were calculated using the iterative methods suggested in [Page et al. 1998]. Real Web users are not random surfers. They are more selective about link-following, they use the Back button to revisit pages and may find new pages through searching rather than browsing. However PageRanks are an indication of likely page popularity. A page with many incoming links and high PageRank, like www.microsoft.com is more likely to be visited than one with few incoming links and low PageRank, like pastime.anu.edu.au. A page with no incoming links and zero PageRank is very unlikely to be found. Search engines rely on "spiders" to crawl the Web, and these too are less likely to find a page if it has fewer incoming links [Lawrence and Giles 1999]. For these reasons, a searching/browsing user is more likely to find a page with more incoming links. However, "popularity" and document utility/relevance, as measured in ad hoc retrieval tasks such as those in TREC, may well be orthogonal.

4.2 Small Web Results

Table 3: Performance of ANU/ACSys Small Web runs using PADRE99 software. Except for the use of PageRank scores, conditions were identical to those prevailing in the Ad Hoc task.

Run-id	Topic Fields	Ave Prec	P@20	Recall	Notes
acsys8wm	TD	.3009	.387	.8231	Content-only
acsys8wmp	TD	.3007	.387	.8213	PageRank wt = 1.0
acsys8wmq	TD	.2804	.3700	.8025	PageRank wt = 10.0
acsys8wmr	TD	.3007	.387	.8213	PageRank wt = 1.0 server bookmarks

Results of Small Web runs are summarised in Table 3.

The minuscule difference in average precision between **acsys8wm** and **acsys8wmp** results from a large number of topics in which PageRanks make no difference at all, and a very small number where they cause harm. The run **acsys8wmr** which used Server bookmarks performed very similarly.

When the normalised PageRanks were scaled up by a factor of 10.0 (run **acsys8wmq**), all three measures were significantly depressed relative to the baseline (by -7%, -4%, and -3% for average precision, precision at 20 and recall). Only topic 413 ("steel production") benefits from use of PageRanks:

TOPIC	AVE PREC.	P@20	RECALL
413	(0.0738 0.0866, +17\%)	(0.1500 0.2000, +33\%)	(0.7500 0.7500, +0\%)

All other topics were either unaffected or were adversely affected.

Further runs were conducted post-hoc to determine the effectiveness of PADRE99 relevance feedback and synonym expansion as applied to the Small Web data. Results are tabulated in Figure 2 but have not yet been statistically analysed. On the surface, results are quite contrary to those obtained on the Ad Hoc collection. Increasing topic length still improves performance but synonym expansion (as implemented)

Average Precision

	plain	-rf	-syn	rf/syn	average
short	0.2678	0.2736	0.3262	0.3242	0.298
medium	0.2999	0.3178	0.3530	0.3525	0.3308
long	0.3117	0.3222	0.3701	0.3663	0.3426
average	0.2931	0.3045	0.3498	0.3477	

Precision @ 20

	plain	-rf	-syn	rf/syn	average
short	0.4890	0.4910	0.5180	0.5150	0.5033
medium	0.5480	0.5530	0.5740	0.5640	0.5598
long	0.5690	0.5580	0.6020	0.5790	0.577
average	0.5353	0.534	0.5647	0.5527	

Topic-by-topic recall

	plain	-rf	-syn	rf/syn	average
short	0.7085	0.7257	0.7931	0.7806	0.752
medium	0.7428	0.7693	0.7976	0.8207	0.7826
long	0.7575	0.7785	0.8093	0.8244	0.7924
average	0.7363	0.7578	0.8	0.8086	

Figure 2: Pre-deadline runs with PADRE99 on the Small Web Task, exploring the effectiveness of synonym expansion and relevance feedback for each topic length

was clearly beneficial while relevance feedback was almost useless by itself and harmful when combined with synonym expansion.

5 Large Web Task

PADRE99 includes various engineering improvements to reduce the impact of indexing and query processing on the virtual memory system. These were sufficient to allow indexing of the 18.5 million page, 100 gigabyte VLC2 collection in under 10 hours on a single 450MHz Pentium 3 system with 1gB of RAM. The official runs relate to an index built as eleven separate components covering approximately 9 gigabytes of text each. The index included all unstemmed words comprising letters only up to a maximum of 12 characters. Each posting recorded the *tf* value for a term,document pair. Position information was not included.

Subsequently, the data was re-indexed in four chunks of approximately 25 gigabytes each.

Each of the 10,000 Large Web Task queries was processed as follows:

1. Stopwords from a list of 51 were eliminated. Note that words were not stemmed.
2. The remaining query words were then sorted by increasing *df*, as estimated by the first index component.
3. Terms were processed until the end of the query unless a high frequency word (occurring in more than 5% of documents) was encountered after at least three terms had been processed.
4. Document content-scores were accumulated in a hash-addressed set of accumulators. No more than 100,000 accumulators were permitted to become active.

5.1 Large Web Task Results

Timing results are reported in detail in the Web Track Overview. In general, query processing speed was acceptable at under about 4 seconds elapsed time per query, whether or not PageRanks were used. However, effectiveness was relatively poor, due perhaps to the use of individual words rather than stems (which may bias term *df* weighting as well as failing to discover useful matches) and to the use of too few document accumulators¹ without a sensible ordering of postings within postings lists (as was done last year.) Time did not permit much experimentation or testing.

Table 4: Performance of ANU/ACSys Small Web runs using PADRE99 software. Except for the use of PageRank scores, conditions were identical to those prevailing in the Ad Hoc task. PageRank scores were derived using Universal bookmarks (i.e. every page was bookmarked).

Run-id	Topic Fields	Mod. Ave Prec	P@10	P@20	Notes
acsys81w0	TD	.2352	.3440	.3360	Content-only
acsys81w0_pr1	TD	.2363	.3460	.3360	PageRank wt = 1.0
acsys81w0_pr10	TD	.2231	.3380	.3350	PageRank wt = 10.0

Results of Large Web runs are summarised in Table 4. It is interesting that the measures for modified average precision and for P@10 are numerically higher for the equal-weighted PageRank run although it is not expected that the differences are statistically significant.

¹Once all the available relevance scoring accumulators have been assigned to documents, score contributions for other documents are ignored.

6 Conclusions

The lack of significant difference between the PADRE98 and PADRE99 long topic Automatic Ad Hoc runs suggests that an index built without stopword elimination and without stemming can be used to achieve the same query processing effectiveness, while avoiding loss of potentially useful information. The significant increase in query processing cost is something which needs to be addressed, as is the relatively poor performance of the efficient relevance feedback mechanism.

The incorporation of PageRank scores in rankings in the Large and Small Web tasks produced no benefit. It is concluded that PageRanks are not useful within the TREC context, even when using queries actually taken from Web search engine logs.

The results on the Large Web task indicate that it is quite feasible to index a 100 gigabyte collection of Web documents on a US\$7,000 PC and to process queries at a reasonable rate. The compactness of PADRE99 indexes has permitted the demonstration of query processing over the 100 gigabyte collection using a Dell laptop (266MHz Pentium II processor with 128 MB of RAM) and only 6.5 gB of disk space. Unfortunately, to achieve this has required taking query processing short cuts (avoiding phrases, stemming and query expansion and limiting the number of document accumulators) which cause harm to effectiveness.

Acknowledgements

Bibliography

- BRIN, S. AND PAGE, L. 1998. The anatomy of a large-scale hypertextual Web search engine. In H. ASHMAN AND P. THISTLEWAITE Eds., *Proceedings of the Seventh International World Wide Web Conference*, Volume 30 of *Computer Networks and ISDN Systems. The International Journal of Computer and Telecommunications Networking* (Amsterdam, April 1998), pp. 107–117. Elsevier. Brisbane, Australia.
- HAWKING, D., THISTLEWAITE, P., AND CRASWELL, N. 1997. ANU/ACSys TREC-6 experiments. In E. M. VOORHEES AND D. K. HARMAN Eds., *Proceedings of the Sixth Text Retrieval Conference (TREC-6)* (Gaithersburg MD, November 1997), pp. 275–290. U.S. National Institute of Standards and Technology. NIST special publication 500-240.
- LAWRENCE, S. AND GILES, C. L. 1999. Accessibility of information on the web. *Nature* 400, 107–109.
- PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. 1998. The pagerank citation ranking: Bringing order to the web. Technical report (January), Stanford, Santa Barbara, CA 93106. <http://www-db.stanford.edu/~backrub/pageranksub.ps>.
- ROBERTSON, S. E., WALKER, S., HANCOCK-BEAULIEU, M., AND GATFORD, M. 1994. Okapi at TREC-3. In D. K. HARMAN Ed., *Proceedings of the Third Text Retrieval Conference (TREC-3)* (Gaithersburg MD, November 1994). U.S. National Institute of Standards and Technology. NIST special publication 500-225.
- SINGHAL, A., SALTON, G., MITRA, M., AND BUCKLEY, C. 1995. Document length normalization. Technical Report TR95-1529, Department of Computer Science, Cornell University, Ithaca NY.

AT&T at TREC-8

Amit Singhal, Steve Abney, Michiel Bacchiani,
Michael Collins, Donald Hindle, Fernando Pereira

AT&T Labs-Research
{singhal,abney,bacchiani,mcollins,hindle,pereira}@research.att.com

Abstract

In 1999, AT&T participated in the ad-hoc task and the Question Answering (QA), Spoken Document Retrieval (SDR), and Web tracks. Most of our effort for TREC-8 focused on the QA and SDR tracks. Results from SDR track show that our document expansion techniques, presented in [8, 9], are very effective for speech retrieval. The results for question answering are also encouraging. Our system designed in a relatively short period for this task can find the correct answer for about 45% of the user questions. This is specially good given the fact that our system extracts only a short phrase as an answer.

1 Introduction

Question answering and spoken document retrieval were the main areas of interest for AT&T at TREC-8. For most of our work, we used an internally modified version of the SMART retrieval system developed at Cornell University [2, 6].

Our question-answering system first retrieves the top ranked passages in response to a user question. These passages are then passed to a linguistic processing sub-system that analyzes the user question and the passages to spot entities that might answer the question. These entities are then ranked based on several heuristics that were developed on training questions. To establish a baseline, we also submitted two runs based only on passage retrieval that did not use the linguistic processing sub-system of our QA system.

For speech retrieval, we continued our experiments with document expansion using related corpora [8, 9]. Results are once again consistently good.

2 Ad-hoc Runs

Our ad-hoc runs are little changed from our 1998 ad-hoc submissions. Please refer to [8] for the details of the algorithms used.

For this task, we strongly believe that full-length TREC topics are artificially long in comparison to real user queries. Another artificiality of the TREC environment is the structure of these topics, *i.e.* the existence of separate **title**, **description** and the **narrative** sections. Such structure will not be encountered in more popular search environments. Therefore, we ignore the **narrative** section in all our runs. Additionally, we ignore the knowledge that certain words are **title** words or **description** words in a topic. We experiment with very short, **title only** queries (**t**), and longer, **title and description** (**t+d**) queries.

We submitted two runs in each category: **att99tc** and **att99te** for **title only** queries and **att99tdc** and **att99tde** for **title and description** queries. Our conservative runs **att99tc** and **att99tdc** are a repeat of our 1998 conservative collection enrichment based runs based on pseudo-feedback. Our experimental runs **att99te** and **att99tde** do some “locality-based” document selection to be used in pseudo-feedback. For these runs, we retrieve the top 50 documents using our standard vector-space ranking. We then re-ranked these fifty documents to promote documents that contain multiple query words in the same sentence or adjoining sentences (see details in Section 3 on Question Answering). Top ten documents from this reranked list are assumed relevant for pseudo-feedback.

Query Sections	Baseline <i>dnb.dtn</i>	Expansion from		Conservative Collection Enrichment
		target collection	TREC D12345	
t (att99atc)	0.2363	0.2736 (+15.8%)	0.2884 (+22.1%)	0.2853 (+20.8%)
# Q better/worse		0/0	29/21	34/16
t (att99ate)	0.2363	0.2740 (+16.0%)	0.2840 (+20.2%)	0.2835 (+20.0%)
# Q better/worse		0/0	24/26	28/22
t+d (att99atdc)	0.2592	0.2943 (+13.5%)	0.3170 (+22.3%)	0.3089 (+19.2%)
# Q better/worse		0/0	30/20	34/16
t+d (att99atde)	0.2592	0.3024 (+16.7%)	0.3237 (+24.9%)	0.3165 (+22.1%)
# Q better/worse		0/0	27/23	29/21

Table 1: Effect of conservative collection enrichment

Run	Average Precision	Best	>= Median	< Median
att99atc (title only)	0.2853	2	32	16
att99ate (title only)	0.2835	4	30	16
att99atdc (title+desc)	0.3089	2	41	7
att99atde (title+desc)	0.3165	4	38	8

Table 2: Results for adhoc runs

The results for our ad-hoc runs are shown in Tables 1 and 2. There are several possible variations on the database used for pseudo-feedback: expansion from the target collection (no collection enrichment), expansion from a large collection (collection enrichment), and conservative collection enrichment [8]. Even though the conservative method has somewhat poorer average precision than methods that expand from the large collection alone (collection enrichment), as the rows labeled # Q better/worse show, it is more stable with respect to the number of queries that improve or deteriorate in comparison to expansion from the target collection (This is the sensible baseline for this comparison since if we compare to unexpanded queries, which is the baseline used in the average precision rows, all expansion strategies will show large gains and the relative performance of different expansions will be hard to judge).

Overall, these results are quite reasonable. In our view, the minor improvements that we gain out of doing various modifications to our simple two-pass pseudo-feedback based algorithm (used in **att99atc** and **att99atdc**), are not fundamentally better and they come at an increased processing cost. Algorithms quite similar to the one used in **att99atdc** have consistently been one of the best over the last three or four TREC conferences. This leads us to wonder whether we are learning something new from the ad-hoc runs in recent years.

3 Question Answering

Our question answering system is a hybrid IR-linguistic system. The retrieval component first retrieves top ranked passages for a question, and the linguistic component then processes those passages in light of the user question to identify entities that can potentially answer the question.

3.1 Passage Retrieval

The passage retrieval system involves the following steps:

1. The top fifty documents for a question are retrieved using a straight vector match (no query expansion).
2. Each section of these top fifty documents is broken into sentences and each sentence is assigned a score based on the following algorithm:
 - the sentence score is initialized to zero, and the passage size is also initialized to zero;

- the query term weight of every question word that appears in the sentence is added to the sentence score, the passage size is set to the sentence size (in bytes);
 - if a query word bigram appears in the sentence, an extra credit¹ is assigned to the sentence,
 - if an adjoining sentence contains a question word not contained in this sentence, and if by adding this adjoining sentence to the passages the passage size doesn't exceed 500 bytes, half the query term weight for this word is added to the sentence score;
 - if a next to adjoining sentence contains a question word not covered yet, and if by adding this adjoining sentence to the passages the passage size doesn't exceed 500 bytes, quarter of the query term weight for this word is added to the sentence score.
3. Highest scoring passage from each section is printed along with its score.
 4. The highest scoring fifty passages are then selected for processing by the linguistic module.

3.2 Linguistic Processing

This section describes the natural-language processing component of the system. Input to this component is the text of the query, together with a set of (ranked) passages that are the output from the IR component. Output from this component is a list of answers ranked in order of importance. The top 5 answers were submitted to the TREC evaluation.

The basic strategy can be divided into three stages:

1. Extract a candidate set of possible answers from the passages, along with their types. The candidate set is a set of entities falling into a number of categories, including people, locations, organizations, quantities, dates, and linear measures. A full list of the types, and a description of how they are extracted, is given in section 3.2.1.
2. Produce a partial ranking of the entities according to how well their type matches the query. Usually this will involve a binary distinction of whether or not an entity is of the correct type: for example, given the query

Who is the author of the book, *The Iron Lady: A Biography of Margaret Thatcher*?

it can be assumed that entities of type **PERSON** are preferred, and a partial ranking is formed where all entities of type **PERSON** are placed ahead of other entity types.

3. Produce a final ordering of the entities by taking into account their frequency and position in the passages. The partial ordering from stage 2 contains many equally ranked entities: in the above example all **PERSON** entities would be ranked the same. The third stage produces a finer grained ranking of entities of the same type through the use of frequency and position information.

The following sections describe in detail how these three processing stages were carried out.

3.2.1 Entity Extraction

The following types of entities were extracted as potential answers to queries:

Proper Names Proper names (capitalized sequences of words) were extracted from the passages, and then classified into one of the categories **PERSON**, **LOCATION**, **ORGANIZATION** or **OTHER** using a classifier built using the method described in [4].²

¹ $0.25 \times (\text{lower of the two component query term weights})$

² The classifier makes a three way distinction between **PERSON**, **LOCATION** and **ORGANIZATION**; names where the classifier makes no decision were classified as **OTHER**.

Dates All years (4 digit numbers starting with 1... or 20...) were extracted from the passages. The CASS parser [1] was used to extract full dates (such as *January 1st 2000*).

Quantities Bare numbers were extracted using the CASS parser. Noun phrases involving modification by a number were also extracted by CASS: for example *The Three stooges*, *4 airports*, *270 people*. In this latter case the headword of the noun phrase (*stooges*, *airports* or *people*) was extracted; these entities could then be later identified as good answers to *How many ...* questions such as *How many stooges were there?*

Durations CASS was used to extract time durations such as *three years*, *four hours* and so on.

Linear Measures CASS was used to extract measure amounts such as *170 miles* or *180 feet*.

We should note that this list of types is almost certainly not complete. Monetary amounts (e.g., *\$25 million*) were added to the system shortly after the TREC run, but other gaps in coverage remain.

3.2.2 Ranking of Entities by Type

This stage involved processing the query to identify the type that is required by the user. The following rules were used to do this:

- All queries starting with *Who*, *Whom* were taken to be of type **PERSON**.
- All queries starting with *Where*, *Whence*, *Whither* were taken to be of type **LOCATION**.
- All queries starting with *When* were taken to be of type **DATE**.
- All queries starting with *How few*, *How great*, *How little*, *How many*, *How much* were taken to be of type **QUANTITY**.
- All queries starting with *How long* were taken to be ambiguous between **DURATION** and **LINEAR_MEASURE**. All queries starting with *How tall*, *How wide*, *How high*, *How big*, *How far* were taken to be **LINEAR_MEASURE**.
- Queries containing the wh-words *Which* or *What* typically also involve a head noun that describes the type of entity involved. These questions fall into two formats: *What X* where *X* is the noun involved, or *What is the ... X*. Here are a couple of examples:

What company is the largest Japanese ship builder?

What is the largest city in Germany?

For these queries the head noun (e.g., *company* or *city*) was extracted, and a lexicon mapping nouns to types was used to identify the type of the query. The lexicon was partly hand-built (including some common cases such as *number* → **QUANTITY** or *year* → **DATE**). A large list of nouns indicating **PERSON**, **LOCATION** or **ORGANIZATION** categories was automatically taken from the contextual (appositive) cues learned in the named entity classifier described in [4].

- In queries containing no wh-word (e.g., *Name the largest city in Germany*), the first noun phrase that is an immediate constituent of the matrix sentence is extracted, and its head is used to determine query type, as for *What X* questions.

If these rules fail to identify the type of the query, then all entities get equal ranking.

In most cases, the query classification stage implies a binary distinction (ranking) of the entities, depending on whether they are or are not of the correct type. However, there are a couple of special cases where finer distinctions are made. If a question is of the **DATE** type, and the query contains one of the words *day*, *month* or *year*, then “full” dates are ranked above years. Conversely, if the query contains the word *year*, then years are ranked above full dates. In *How many X* questions (where *X* is a noun), quantified phrases whose head noun is also *X* are ranked above bare numbers of other quantified phrases: for example, in the query *How many lives were lost in the Lockerbie air crash*, entities such as *270 lives* or *almost 300 lives* would be ranked above entities such as *270 people* or *150*.

Run	Mean Answer Length	Answer in Top-5	Mean Score
attqa50e	10.5 bytes	89/198	0.356
attqa50p	50 bytes	77/198	0.261

Table 3: Results for the 50-byte answer category

3.2.3 Ranking of Entities by Position

Finally, the frequency and position of entities in the retrieved passages was taken into account. First, some normalization of entities is done: dates are mapped to the format year-month-day, proper names are normalized by last-name, Then a score is calculated for each entity. Each time an entity occurs in the top-ranked passage (or passages) from the IR system, its score is incremented by 10. Each time it occurs in a lower ranked passage, its score is incremented by 1. This score is used as a secondary ranking method superimposed on the partial ranking given by the query classifier.

3.3 Results

We submitted two linguistics based runs, one in the 50-byte answer category and another in the 250-byte category. We also submitted two comparable passage retrieval based runs.

3.3.1 50-bytes

The most realistic run is our 50-byte entity based run **attqa50e**; this run extracts only the entity that the system thinks is *the answer*. The details of this run are described above in the section on linguistic processing. For comparison we also did a 50-byte passage based run (which involves no entity recognition/extraction). The passage only run trims the top ranked sentences to reduce them down to fifty bytes. It drops some function words from a sentence, and it drops the question words assuming they won't be in the answer. If the resulting trimmed sentence is still over fifty bytes, it outputs the first 50 characters.

The results from our 50 bytes runs: **attqa50e**, the entity-based run, and **attqa50p**, the passage-based run, are shown in Table 3. We expected our entity based run to be better than the passage based run, and it is. This reinforces the belief that IR system need context to do their job well. When an IR system is constrained to extract a very tiny piece of text as *the answer*, it doesn't do very well. Even with extracting about five times as much text as compared to our entity-based system (50 bytes/answer instead of just 10.5 for the entity-based system), the passage-based system gets fewer answers right and the answers are not ranked well. These results indicate that to do question answering such that the answer is just a phrase (or a very short snippet of text) we will need to enhance a purely keyword-based system with some linguistic processing.

3.3.2 250-bytes

Our 250-bytes passage based run **attqa250p** involves the following steps:

1. The passages retrieved as per the algorithm described in Section 3.1 are first refined so that no single document contributes more than one passage to the passage pool. (As described in Section 3.1, different sections of a document can each contribute a passage to the passage pool.) When one document supplies more than one passage, the highest scoring passage is selected. Ties are broken in favor of longer passages as they have a higher chance of containing the answer.
2. Near-duplicate passages are removed from the pool. If a low-scoring passage has a cosine-similarity of over 0.50 with a highly ranked passage, the low-scoring passage is removed from the pool. The main motivation behind doing this is to improve our chances of hitting the answer in one of the top five passages, instead of repeating same information in multiple passages.

Run	Mean Answer Length	Answer in Top-5	Mean Score
attqa250p	249 bytes	135/198	0.545
attqa250e	247 bytes	120/198	0.483

Table 4: Results for the 50-byte answer category

3. The top five passages from the remaining pool are printed in order of their scores. If a passage is longer than 250 bytes, the *key-sentence* of that passage (remember we build passages around a key sentence by adding previous and next sentences, and we keep adding sentences as long as we are under 500 bytes) is printed. If we still have some bytes to spare, the later bytes of previous sentence are added, and then the earlier bytes of the later sentence.

Our 250-bytes linguistic (entity) based run **attqa250e** involves the following steps:

1. The ranked list of entities as ranked for our entity-based run **attqa50e** is used as a starting point.
2. Passages are ranked using the passage ranking algorithm described in Section 3.1.
3. The first passage that contains the top-ranked entity in the entity list is presented to the user. The top ranked entity and other entities covered by this passage are removed from the entity list.
4. This process is repeated until we have five 250 bytes passages.

Results for our 250-byte runs are shown in Table 4. Our passage based run in the 250-bytes category was one of the best runs in this track. This result indicates that when a passage-retrieval system is allowed some more text in its output, it can do a very good job answering questions. This further reinforces the belief that IR systems need context to do their job well. We believe that these results can be improved notably with more effort, and we are working in that direction.

3.4 Error Analysis of the Entity-Based System

3.4.1 Ranking of Answers

We looked first at the performance of the entity-based system, considering the queries where the correct answer was found somewhere in the top 5 answers (46% of the 198 questions). We found that on these questions, the percentage of answers ranked 1, 2, 3, 4, and 5 was 66%, 14%, 11%, 4%, and 4% respectively. This distribution is by no means uniform; it is clear that when the answer is somewhere in the top five, it is very likely to be ranked 1st or 2nd. The system's performance is thus bimodal: it either completely fails to get the answer, or else recovers it with a high rank.

3.4.2 Accuracy on Different Categories

Table 5 shows the distribution of question types in the TREC-8 test set ("Percentage of Questions"), and the performance of the entity-based system by question type ("System Accuracy"). We categorized the questions by hand, using the eight categories described in section 3.2.1, plus two categories that essentially represent types that were not handled by the system at the time of the TREC deadline: **Monetary Amount** and **Miscellaneous**.

"System Accuracy" means the percentage of questions for which the correct answer was in the top five returned by the system. There is a sharp division in the performance on different question types. The categories **Person**, **Location**, **Date** and **Quantity** are handled fairly well, with the correct answer appearing in the top five 60% of the time. These four categories make up 67% of all questions. In contrast, the other question types, accounting for 33% of the questions, are handled with only 15% accuracy.

Unsurprisingly, the **Miscellaneous** and **Other Named Entity** categories are problematic; unfortunately, they are also rather frequent. Table 6 shows some examples of these queries. They include a large tail of

Type	Percentage of Questions	System Accuracy	Type	Percentage of Questions	System Accuracy
Person	28	62.5	Other Named Entity	14.5	31
Location	18.5	67.6	Miscellaneous	8.5	5.9
Date	11	45.5	Linear Measure	3.5	0
Quantity	9.5	52.7	Monetary Amount	3	0
			Organization	2	0
			Duration	1.5	0
TOTAL	67	60	TOTAL	33	15

Table 5: Performance of the entity-based system on different question types. "System Accuracy" means percent of questions for which the correct answer was in the top five returned by the system. "Good" types are on the left, "Bad" types are on the right.

What does the Peugeot company manufacture?
Why did David Koresh ask the FBI for a word processor?
What are the Valdez Principles?
What was the target rate for M3 growth in 1992?
What does El Nino mean in spanish?

Table 6: Examples of "Other Named Entity" and "Miscellaneous" questions.

questions seeking other entity types (mountain ranges, growth rates, films, etc.) and questions whose answer is not even an entity (e.g., "Why did David Koresh ask the FBI for a word processor?")

For reference, Table 7 gives an impression of the sorts of questions that the system does well on (correct answer in top five).

3.4.3 Errors by Component

Finally, we performed an analysis to gauge which components represent performance bottlenecks in the current system. We examined system logs for a 50-question sample, and made a judgment of what caused the error, when there was an error. Table 8 gives the breakdown. Each question was assigned to exactly one line of the table.

The largest body of errors, accounting for 18% of the questions, are those that are due to unhandled types, of which half are monetary amounts. (Questions with non-entity answers account for another 4%.) Another large block (16%) is due to the passage retrieval component: the correct answer was not present in the retrieved passages. The linguistic components together account for the remaining 14% of error, spread evenly among them.

Question	Rank	Output from System
Who is the author of the book, The Iron Lady: A Biography of Margaret Thatcher?	2	Hugo Young
What is the name of the managing director of Apricot Computer?	1	Dr Peter Horne
What country is the biggest producer of tungsten?	1	China
Who was the first Taiwanese President?	1	Taiwanese President Li Teng hui
When did Nixon visit China?	1	1972
How many calories are there in a Big Mac?	4	562 calories
What is the acronym for the rating system for air conditioner efficiency?	1	EER

Table 7: A few TREC questions answered correctly by the system.

Errors	
Passage retrieval failed	16%
Answer is not an entity	4%
Answer of unhandled type: money	10%
Answer of unhandled type: misc	8%
Entity extraction failed	2%
Entity classification failed	4%
Query classification failed	4%
Entity ranking failed	4%
Successes	
Answer at Rank 2-5	16%
Answer at Rank 1	32%

Table 8: Breakdown of questions by error type, in particular, by component responsible. Numbers are percent of questions in a 50-question sample.

The cases in which the correct answer is in the top five, but not at rank one, are almost all due to failures of entity ranking.³ Various factors contributing to misrankings are the heavy weighting assigned to answers in the top-ranked passage, the failure to adjust frequencies by “complexity” (e.g., it is significant if *22.5 million* occurs several times, but not if *3* occurs several times), and the failure of the system to consider the linguistic context in which entities appear.

4 SDR Runs

We used our own speech recognizer to process the SDR track data. In this track, we continued our experimentation with document expansion from last year [8, 9]. This year we only submitted runs based on document expansion. Our first run **att-s1** is a reproduction of the algorithm we developed in [9]. Our second run **att-s2** is also based on document expansion, but it is aimed at containing excessive increase in weights of already important document terms (see below for details).

4.1 Speech Recognizer

The speech recognition system used for the SDR track used a multi-pass search paradigm. The resulting transcriptions were obtained by performing two recognition passes, the first using both an acoustic and low complexity language model, the second retaining the acoustic scores from the first pass and using a more complex language model. The acoustic model of this system is described in section 4.1.1, the language models and search algorithm are described in section 4.1.2.

4.1.1 Acoustic model

The acoustic model was trained on all the SDR track data available from previous evaluations. The data used for training was from 14 different news programs from the period May 10, 1996 until January 31, 1998. The total amount of transcribed recordings used in training was 143 hours.

The speech waveforms were parameterized using a mel-frequency cepstral analysis and energy measurements. The system used the first 12 mel-frequency cepstral coefficients and a normalized energy parameter as well as the first and second derivatives (39 dimensions in total), computed at a rate of 100 frames per second. To compensate for channel effects, the cepstral mean of the signal was subtracted for the cepstral vectors.

³The sole exception was a query misclassification caused by a parse failure—miraculously, the correct answer made it to rank five despite being of the “wrong” type.

A training dictionary for all 36475 unique words seen in the training transcriptions was generated using our text-to-speech system [3] followed by hand corrections/additions. The resulting dictionary had 38616 entries (average of 1.06 entries per unique word). The used phone set consisted of 42 phone models, 1 silence model and 5 non-speech models.

All phones were modeled using three-state left-to-right HMMs except for the silence model which was a single state HMM. All state emission distributions were modeled by Gaussian mixture densities. Mixture densities were estimated by iteratively segmenting the data using the Viterbi algorithm and estimating mixture densities for the given segmentation using the Expectation Maximization (EM) algorithm (i.e. the mixture identities were hidden but the state segmentations were not). To initialize the mixture components for the EM algorithm, the k -means clustering algorithm with a Euclidean distance metric was used on variance normalized data. The final acoustic model was trained in three stages. In the first stage, a context independent system was built. To bootstrap this first training stage, an initial state-level segmentation was obtained by a Viterbi alignment using our last evaluation system. Then 20 mixture component state emission densities were estimated in three iterations. In the first iteration, 8 mixture component densities were estimated. In the second, the number of mixture components was increased to 20 and this model was refined by another iteration. In the second training stage, the sharing among triphone state emission distributions were defined. Shared state distributions were defined by decision tree clustering using a likelihood design criterion and allowing questions about the phonetic identity of the phone context. Finally, in the third stage, mixture densities were estimated for the shared state distributions. The final densities were obtained in four iterations, the first two to estimate 4 mixture component densities, the second two to estimate 12 mixture component densities.

4.1.2 Language model and Search Algorithm

In the first recognition pass, lattices were built that were rescored in a second recognition pass. The most likely transcripts were then used together with the acoustic model from the first pass to find the boundary times of the words in the transcriptions by the Viterbi algorithm.

The first recognition pass used a pruned trigram language model, the second an un-pruned 6-gram model. Both first and second pass models were Katz [5] backoff language models. The first pass trigram model was pruned using the approach of Seymore and Rosenfeld [7] using a pruning threshold of 100. In addition to the transcriptions of previous SDR evaluations we also used the transcripts of the Hub4 evaluations and two printed media sources (the LDC North American news corpus and United Press International (ClariNet)).

Different language models were constructed for every two week period in the evaluation data. A total of 11 sets of first and second pass language models were constructed for the 5 month period that the evaluation data covers. First a model was constructed for the first two week period using all available training data prior to that period. Then, for the construction of the models for each subsequent two week period, the data from the preceding two week period was added to the data used for the first two week period model. A weighting scheme was used to emphasize the contribution of the most recent two week period data as well as to emphasize the spoken news transcripts with respect to the printed sources. To accompany the two week language models, a different dictionary was used for each two week period. The dictionaries included all unique words found in the transcriptions as well as all unique words occurring with a frequency larger than two in the printed sources. The sizes of the dictionaries for the different two week periods ranged from 210340 entries to 261215 entries.

4.2 Retrieval System

We used the NA News corpus and UPI news (also used in the language model training described above) as the related corpus for document expansion as well as the large collection for conservative collection enrichment (see [8]) for query expansion. The retrieval cutoff date for this track was July 1, 1998. For 1998, the NA News corpus only has news for January to April 1998. We use all these news articles in our runs. We also added to this UPI news available through Clarinet news for the months of April to June 1998. This gave us a related corpus of 182,755 news articles.

Algorithm/Transcript details	No query expansion	Query expansion from		Conservative Coll. Enrich.
		target collection	Print News	
No document expansion ltt (Closed Captions)	0.4574 —	0.5103 +11.6%	0.5742 +25.5%	0.5390 +17.8%
No document expansion nist-b1 (WER:27.5%)	0.4113 —	0.4888 +18.8%	0.5498 +33.7%	0.5194 +26.3%
Loss due to ASR	(-10.1%)	(-4.2%)	(-4.2%)	(-3.6%)
No document expansion att-s1 (WER:29.3%)	0.4058 —	0.4798 +18.2%	0.5506 +35.7%	0.5164 +27.3%
Loss due to ASR	(-11.3%)	(-6.0%)	(-4.1%)	(-4.2%)
No document expansion cmu-s1 (WER:64.4%)	0.2916 —	0.3740 +28.2%	0.4123 +41.4%	0.3970 +36.1%
Loss due to ASR	(-36.3%)	(-26.7%)	(-28.2%)	(-26.3%)
No document expansion cuhtk-s1 (WER:20.5%)	0.4286 —	0.5055 +17.9%	0.5667 +32.2%	0.5339 +24.6%
Loss due to ASR	(-6.3%)	(-1.0%)	(-1.3%)	(-0.9%)
No document expansion cuhtk-slp1 (WER:26.6%)	0.4233 —	0.4890 +17.9%	0.5531 +32.2%	0.5212 +24.6%
Loss due to ASR	(-7.5%)	(-4.2%)	(-3.7%)	(-3.3%)
No document expansion limsi-s1 (WER:21.5%)	0.4226 —	0.5014 +18.7%	0.5554 +31.4%	0.5224 +23.6%
Loss due to ASR	(-7.6%)	(-1.7%)	(-3.3%)	(-3.1%)
No document expansion shef-s1 (WER:32.0%)	0.4001 —	0.4770 +19.2%	0.5402 +35.0%	0.5065 +26.6%
Loss due to ASR	(-12.5%)	(-6.5%)	(-5.9%)	(-6.0%)

Table 9: SDR Runs: No Document Expansion.

Algorithm-1

Our first document expansion algorithm is taken verbatim from our previous work presented in [9]. The query expansion algorithm is the same as the one used in our ad-hoc runs, only the target collection and the large collection for conservative collection enrichment are different. The target collection is the SDR collection and the large collection is the NA News and UPI news collection described above.

The results for our SDR runs are shown in Tables 9–11. Here are the main observations from these results.

1. Speech retrieval over automatically recognized speech is very viable. For reasonable transcripts, the losses in retrieval effectiveness are minimal, 1–5% (the negative numbers shown in parentheses).
2. As expected, query expansion via pseudo-feedback is useful across the board. This can be observed in the last three columns of the Tables. In each of these columns, the second entry shows the improvements of the corresponding query expansion algorithm over no query expansion.
3. Our conservative query expansion hurt us in this environment. This is evident by the consistently better results from doing query expansion from the print news vs. doing conservative collection enrichment. For example, when doing retrieval from closed caption (second row in Table 10), doing query expansion from print news yields an average precision of 0.5742, whereas our conservative query expansion yields only 0.5390, a noticeable drop.
4. Document expansion (see Table 10) is consistently beneficial. For example, our reference run **att-r1** would have been 0.5390 instead of 0.5600 had we not used document expansion.
5. Retrieval effectiveness is not very sensitive to WER of the recognizers for reasonable recognition. This is evident by looking at all our official runs (other than the one on cmu's transcripts) which have

Algorithm/Transcript details	No query expansion	Query expansion from		Conservative Coll. Enrich.
		target collection	Print News	
Document expansion (Algo-1) ltt (Closed Captions)	0.4918 —	0.5371 +9.2%	0.5804 +18.0%	0.5600 att-r1 +13.8%
Document expansion (Algo-1) nist-b1 (WER:27.5%) Loss due to ASR	0.4779 — (-2.8%)	0.5427 +13.6% (+1.0%)	0.5646 +18.2% (-2.7%)	0.5539 att-b1 +15.9% (-1.0%)
Document expansion (Algo-1) att-s1 (WER:29.3%) Loss due to ASR	0.4639 — (-5.7%)	0.5207 +12.2% (-3.1%)	0.5586 +20.4% (-3.8%)	0.5431 att-s1 +17.1% (-3.0%)
Document expansion (Algo-1) cmu-s1 (WER:64.4%) Loss due to ASR	0.3752 — (-23.7%)	0.4369 +16.5% (-18.7%)	0.4635 +23.5% (-20.2%)	0.4526 att-cr-cmus1 +20.6% (-19.2%)
Document expansion (Algo-1) cuhtk-s1 (WER:20.5%) Loss due to ASR	0.4901 — (-0.3%)	0.5421 +10.6% (+0.9%)	0.5715 +16.6% (-1.5%)	0.5592 att-cr-cuhtks1 +14.1% (-0.1%)
Document expansion (Algo-1) cuhtk-s1p1 (WER:26.6%) Loss due to ASR	0.4724 — (-3.9%)	0.5309 +12.4% (-1.1%)	0.5647 +19.5% (-2.7%)	0.5494 att-cr-cuhtks1p1 +16.3% (-1.9%)
Document expansion (Algo-1) limsi-s1 (WER:21.5%) Loss due to ASR	0.4717 — (-4.1%)	0.5346 +13.3% (-0.5%)	0.5631 +19.4% (-3.0%)	0.5516 att-cr-limsis1 +16.9% (-1.4%)
Document expansion (Algo-1) shef-s1 (WER:32.0%) Loss due to ASR	0.4710 — (-4.2%)	0.5277 +12.0% (-1.8%)	0.5588 +18.6% (-3.7%)	0.5455 att-cr-shefs1 +15.8% (-2.6%)

Table 10: SDR Runs: Document Expansion, Algorithm-1

average precision values in the range for 0.5431 to 0.5600. There is an insignificant 3% gap in average precision between doing retrieval on closed caption vs. doing retrieval on ASR transcripts which have up to 32% WER.

Algorithm-2

We remind you that our term weighting scheme assigns weights to words in a document based on their occurrence frequency in the document and the length of the document. The two basic factor are the *tf*-factor, which accounts for the fact that words that are repeated within a document are more important; and the document length normalization factor which is used to assign lower weights to all words in very long documents. Within one document, the document length normalization factor is same for all terms. However, the *tf*-factor changes from word to word based on the word's frequency. During the last few years, we have realized that a word that appears three times in a document is not thrice as important than a word that appears just once so we have been using a dampened *tf*-factor (a logarithmic or a double-log factor) which rises sub-linearly with the increase in word frequency. In particular, we use the double log *tf*-factor, *i.e.* a word with frequency *tf* gets a weight of $1 + \ln(1 + \ln(tf))$.

During the course of our experiments with document expansion, we noticed that the expansion algorithm we use above tends to create an unwanted imbalance in weights of different document terms. For example, consider a document which contains many instances of the word **information**. When we use this document as a query to find related printed documents, many of the related documents also mention the word **information**, since **information** is an important word in the query vector, *i.e.* the vector for the recognized document. When we do document expansion using Rocchio's formula, the word **information** gets a further boost in its weight and it ends up being a very heavily weighted word for this document. This is contrary to the reason for using a dampened *tf*-factor in our term weighting scheme (as described above).

To address this problem, we changed our document expansion scheme to ensure that frequent words do

Algorithm/Transcript details	No query expansion	Query expansion from		Conservative Coll. Enrich.
		target collection	Print News	
Document expansion (Algo-2) ltt (Closed Captions)	0.4821 —	0.5403 +12.1%	0.5863 +21.6%	0.5677 +17.7%
Document expansion (Algo-2) nist-b1 (WER:27.5%) Loss due to ASR	0.4610 — (-4.4%)	0.5406 +17.3% (+0.1%)	0.5716 +24.0% (-2.5%)	0.5634 +22.2% (-0.8%)
Document expansion (Algo-2) att-s1 (WER:29.3%) Loss due to ASR	0.4536 — (-5.9%)	0.5208 +14.8% (-3.6%)	0.5731 +26.3% (-2.3%)	0.5510 att-s2 +21.5% (-2.9%)
Document expansion (Algo-2) cmu-s1 (WER:64.4%) Loss due to ASR	0.3520 — (-27.0%)	0.4077 +15.8% (-24.6%)	0.4643 +31.9% (-20.8%)	0.4508 +28.1% (-20.6%)
Document expansion (Algo-2) cuhtk-s1 (WER:20.5%) Loss due to ASR	0.4711 — (-2.3%)	0.5496 +16.7% (+1.7%)	0.5829 +23.7% (-0.6%)	0.5705 +21.1% (+0.5%)
Document expansion (Algo-2) cuhtk-s1p1 (WER:26.6%) Loss due to ASR	0.4601 — (-4.6%)	0.5336 +16.0% (-1.2%)	0.5678 +23.4% (-3.2%)	0.5539 +20.4% (-2.4%)
Document expansion (Algo-2) limsi-s1 (WER:21.5%) Loss due to ASR	0.4645 — (-3.7%)	0.5349 +15.2% (-1.0%)	0.5698 +22.7% (-2.8%)	0.5551 +19.5% (-2.2%)
Document expansion (Algo-2) shef-s1 (WER:32.0%) Loss due to ASR	0.4508 — (-6.5%)	0.5229 +16.0% (-3.2%)	0.5598 +24.2% (-4.5%)	0.5452 +21.0% (-3.9%)

Table 11: SDR Runs: Document Expansion, Algorithm-2.

not end up getting very heavy weights. Under this scheme, any word is allowed an increment of *one* in its raw frequency due to expansion. For example, if a word occurred once in the document, its pre-expansion weight was 1.0 (ignoring document length normalization). If the post-expansion weight for this words becomes 2.0, which will correspond to a post-expansion raw frequency of 5.57 (since $1 + \ln(1 + \ln(5.57)) = 2.0$), then this increment is not allowed and the raw frequency increase is capped at 1, yielding the post-expansion raw frequency of 2 and a post-expansion weight of 1.53. This effect is even more visible for words with very high raw frequencies (like 10).

We submitted a second run based on this document expansion scheme and the results are shown in Table 11. Comparing this document expansion algorithm **att-s2** (Algo-2) with our previous algorithm **att-s1** (Algo-1) in Table 10, we do see that this algorithm yields consistently better results than our old algorithm. It in fact yields the best results for every transcription, which are shown in column-4 of Table 11

Run	Average Precision	Best	\geq Median	$<$ Median
att-r1	0.5600	7	27	15
att-b1	0.5539	1	35	13
att-s1	0.5431	5	29	15
att-s2	0.5510	4	31	14
att-cr-cmus1	0.4626	0	18	31
att-cr-cuhtks1	0.5592	2	35	12
att-cr-cuhtks1p1	0.5494	2	32	15
att-cr-limsis1	0.5516	2	32	15
att-cr-shefs1	0.5455	3	30	16

Table 12: Results for SDR runs

Query Sections	Baseline <i>dnb.dtn</i>	Expansion from		Conservative Collection Enrichment
		target collection	TREC D12345	
t+d (att99wtde)	0.2470	0.2876 (+16.5%)	0.3033 (+22.8%)	0.3091 (+25.2%)
t+d (att99wtde)	0.2470	0.2883 (+16.8%)	0.3138 (+27.1%)	0.3113 (+26.0%)

Table 13: Effect of conservative collection enrichment

Run	Average Precision	Best	\geq Median	$<$ Median
att99wtde (title+desc)	0.3033	2	40	8
att99wtde (title+desc)	0.3113	2	37	11

Table 14: Results for adhoc runs

(in boldface). Table 12 presents some other statistics on our official runs.

5 Web Track Runs

We submitted two runs for the small Web task: **att99wtde** and **att99wtde**. These runs correspond to our ad-hoc runs att99atdc and att99atde, respectively. The only difference is that for the web runs, we remove duplicates from the initial list of documents used in pseudo-feedback. These runs are *content-only* runs and do not use the linkage analysis commonly used by Web search engines. For these runs, we first retrieve the top 100 documents using our standard vector-space ranking. If two documents in this list have a cosine similarity over 0.80, we assume they are duplicates of each other and remove the lower ranked document from this list.

For run **att99wtde**, the list of documents obtained above after duplicate removal is further re-ranked using sentence based locality described in our question answering effort. Top 10 documents from this reranked list are assumed relevant and are used in pseudo-feedback based query expansion. For run **att99wtde**, no reranking is done and the top 10 documents are used in pseudo-feedback. Both these runs use the **title** and **description** sections of the queries. TREC disks 1–5 are used as the larger collection for conservative collection enrichment.

The results from our runs on the 2G web data are shown in Tables 13 and 14. These results are reasonable, especially given the fact that we did not change our retrieval algorithm in any significant manner for Web data.

We also submitted two runs for the large Web track: **att99vlci** and **att99vlcm**. Both runs are based on merging results from twenty different collection formed by dividing the 100G Web data into twenty 5G collections. The run **att99vlcm** merges the document frequencies from various collections and updates every collection so that every collection has a uniform view of the global inverse document frequency for terms. The run **att99vlci** ignores these issues and take document scores from various collections at their face value. There is no query expansion used in these runs. These runs are a straight-forward vector-space match between the query and the documents.

The precision in top 10 documents for **att99vlci** is 0.6180 and for **att99vlcm** is 0.5980. These numbers show that if a very large collection is divided into smaller sub-collections, then one can simply ignore the global-idf issues and merge results from the individual sub-collections to get effective ranking.

6 Conclusions

Our SDR work establishes the usefulness of document expansion. We are very happy to see the incorporation of the question answering track in TREC and look forward to our continuous participation in it next year.

Acknowledgments

We are thankful to to Andrej Ljolje and Michael Riley for their help in building the recognizer for the SDR track data.

References

- [1] Steven Abney. Partial parsing via finite-state cascades. *J. Natural Language Engineering*, 2(4):337–344, December 1996.
- [2] Chris Buckley. Implementation of the SMART information retrieval system. Technical Report TR85-686, Department of Computer Science, Cornell University, Ithaca, NY 14853, May 1985.
- [3] C. Coker. A dictionary-intensive letter-to-sound program. *Journal of Acoustical Society America, Supplement 1*, pages 78–87, 1985.
- [4] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *EMNLP*, 1999.
- [5] S.M. Katz. Estimation of probabilities from sparse data from the language model component of a speech recognizer. *IEEE Transactions of Acoustics, Speech and Signal Processing*, pages 400–401, 1987.
- [6] Gerard Salton, editor. *The SMART Retrieval System—Experiments in Automatic Document Retrieval*. Prentice Hall Inc., Englewood Cliffs, NJ, 1971.
- [7] Kristie Seymore and Ronald Rosenfeld. Scalable backoff language models. In *ICSLP'96*, volume 1, 1996.
- [8] Amit Singhal, John Choi, Donald Hindle, David Lewis, and Fernando Pereira. AT&T at TREC-7. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-6)*, pages 239–252, 1999.
- [9] Amit Singhal and Fernando Pereira. Document expansion for speech retrieval. In *Proceedings of the Twenty Second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 34–41. Association for Computing Machinery, New York, August 1999.

CMU Spoken Document Retrieval in Trec-8: Analysis of the role of Term Frequency TF

M. Siegler, R. Jin and A. Hauptmann

The participation of Carnegie Mellon University in the TREC-8 Spoken Document Retrieval Track used the basic same Sphinx speech recognition system as in TREC-7. Due to some unfortunate defaults in the parameter setup files, the speech recognizer did not perform in a reasonable manner. We will not analyze the results of the speech recognizer runs, as we believe the results contained abnormal types of errors, and insights or improvements on these errors would not generalize. A thorough examination of the speech recognition condition is given in [3]. However, we did evaluate a slightly modified weighting scheme in the reference (R1) and baseline (B1) conditions, which is described below.

1. Motivation for the Retrieval Formulas

The formula we used for cmu-r1 and cmu-b1 runs is based on dtb described by Singhal[1]. In addition, to improve the accuracy, we use the standard pseudo-relevance feedback [2], conservative collection enrichment [1] and document expansion [1]. Since what we have done in that part is similar to what AT&T has done in TREC7 [1], we are not repeating the description of this approach.

Here we would like to discuss the modification we make in the dtb formula. As many people pointed out in previous TRECs, directly multiplying term frequency tf with inverse document frequency idf generally causes poor performance. The poor performance seems to be due to an overweighting of the tf term. To avoid this overestimation of tf , researchers have used $\ln(tf+1)$ or even $\ln(\ln(tf)+1)+1$.

However, these approaches look more empirical than theoretical. Our modification on the tf term is a theoretically motivated attempt to resolve this problem.

2. Analysis

Usually idf for a word A is written as $\log((N+1)/M)$. Here N is the total number of documents in the collection and M is number of documents having at least one occurrence of the word A within the collection. In other words, idf for word A can be thought of as $-\log(p)$ where p is the probability that a document contains at least one occurrence of word A in the collection. Then the term $tf*idf$ for a word A can be written as

$$Tf * idf = -tf * \log(p) = -\log(ptf). \quad (1)$$

We can easily extend the meaning of idf to interpret the term $tf*idf$ for word A as $-\log(p')$ and p' is the probability that a document contains at least tf occurrences of word A . So we have

$$Tf * idf = -\log(p) \quad (2)$$

Combing the two formulas together, we have
 $p' = ptf$,

which means that the probability that a document contains at least tf occurrences of the word A is the probability that a document contains at least one occurrence of the word A to the power tf . Obviously this can be true only when the independent assumption that one occurrence of the word A has nothing to do with another occurrence of the word A is correct.

However, because of the complicated correlation within word occurrences, the independent assumption generally is wrong and will cause underestimation of probability p' . We think this is the reason why multiplying tf with idf directly generally causes overestimation and gives rise to poor performance.

3. Solution

To avoid this problem of overestimation by multiplying tf directly, we have come up with two solutions to replace $tf*idf$. Since $tf*idf$ for word A can be interpreted as $-\log(p')$ and p' is the probability that a document contains at least tf occurrences of word A , the key issue is how to estimate this probability p' .

One solution is using the word histogram directly. For each word A , we can build up a histogram function $N(x, A)$ that tells the number of documents containing exact x occurrences of the word A . With this histogram function, we can compute the " $tf*idf$ " for word A as

$\log((N+1)/G(tf, A))$.

Here $G(tf, A)$ is defined as

$G(tf, A) = \text{Sum}(N(x, A))$ over x and x is integer from x to infinity.

The second method uses a fitted Gaussian distribution to estimate the probability p' . For each word A , we can compute the average occurrences of word A avg_A and standard deviation of occurrences of word A std_dev_A from the histogram function $N(x, A)$. Now we can build the normalized Gaussian distribution as $D(x, avg_A, std_dev_A)$.

Then the " $tf*idf$ " can be computed as

$-\log(l(tf, A))$.

Here $l(tf, A)$ is defined as

$l(tf, A) = \text{Integral of } D(x, avg_A, std_dev_A) \text{ over } x \text{ and } x \text{ is from } tf \text{ to infinity.}$

Furthermore we can use the standard error function to represent $l(tf, A)$ as the following:

```
0.5 * err((tf - avg_A)/sqrt(2)/std_dev_A)
if tf >= avg_A
l(tf, A) =
0.5 * err((avg_A - tf)/sqrt(2)/std_dev_A) + 0.5
if tf < avg_A
```

Both these two approaches have their advantages and disadvantages. The good side of first approach is that it uses the exact data and makes no assumption or approximation. However it may be misled by the local fluctuation. As for the second approach, it complements the down side of the first approach by using fitted Gaussian distribution. However it may cause disaster if the data doesn't fit in Gaussian distribution or when a small data set doesn't reliably estimate the true average and standard deviation.

To obtain the good properties of both approaches, we created a new method. We modify the histogram function $N(x, A)$ for word A as follows: Instead of using natural granularity 1 for x , we use standard deviation $\text{std_dev_}A$ as the granularity. We define the granularity function for each word A as

$$\text{grand}(A) = \text{MAX}(\text{std_dev_}A/3, 1).$$

Now we can define a new histogram function $N'(x, A)$ as

$$N'(x, A) = \text{Sum}(N(y, A)) \text{ over } y \text{ and } y \text{ is from } \text{floor}(x / \text{grand}(A)) * \text{grand}(A) \text{ to } \text{ceiling}(x / \text{grand}(A)) * \text{grand}(A).$$

From the definition of $N'(x, A)$, it is easily seen that $N'(x, A)$ is the same for all x in the range $[\text{floor}(x / \text{grand}(A)) * \text{grand}(A) .. \text{ceiling}(x / \text{grand}(A)) * \text{grand}(A)]$.

Now we can use the first approach to compute the "tf*idf" except that this time the histogram function is $N'(x, A)$ instead of $N(x, A)$. Since the new histogram function is defined as a sum of the old histogram function over an interval on the order of standard deviation, it will be more stable and avoids some risks of the first approach.

4. Experiment

From experiments we have performed on the TREC data, we find out that the approach described above for computing factor "tf*idf" is better than $\text{tf} * \text{idf}$ or $(\ln(\text{tf}) + 1) * \text{idf}$. However, we did not see any significant improvement in performance of our formula over $(\ln(\ln(\text{tf}) + 1) + 1) * \text{idf}$. Instead our formula is generally slightly worse than the $(\ln(\ln(\text{tf}) + 1) + 1) * \text{idf}$ factor.

By comparing documents weighted by our schema and weighted by $(\ln(\ln(\text{tf}) + 1) + 1) * \text{idf}$, we find out that our schema still has the problem of overestimating tf especially when the tf is larger. We think it is due to the fact that when tf is close to the largest tf , the $G(\text{tf}, A)$ is very inaccurate because of the lack of histogram data. In the future we will introduce special treatment for this "ending effect".

5. Conclusion

In this paper we attempted tf*idf a more theoretic interpretation and point out a possible reason why multiplying tf directly with idf causes the poor performance with the given interpretation. We came up with a word histogram based method of integrating tf and idf into one factor which is $\log(1/p')$ and p' is the probability that a document has at least tf occurrences of a particular word. We have several success over tf*idf and $(\ln(\text{tf})+1)*\text{idf}$ and fail to compete with $(\ln(\ln(\text{tf})+1)+1)*\text{idf}$. We think the failure is due to the "ending effect" and we will pursue the problem further in the future.

6. References

- [1] Singhal.A. AT&T at TREC7. In E. M. Voorhees and D. K. Harman, editors, TEXT RETRIEVAL CONFERENCE (TREC-7), page 141-151, 1998.
- [2] J.J Rocchio. Relevance feedback in information retrieval. In The SMART Retrieval System-Experiments in Automatic Document Processing, page 313-323, Englewood Cliffs, NJ, 1971. Prentice Hall, Inc.
- [3] Siegler,M. Integration of Continuous Speech Recognition and Information Retrieval for Mutually Optimal Performance, PhD. Thesis, Electrical and Computer Engineering, Carnegie Mellon University, 1999
<http://www.cs.cmu.edu/~msiegler/publish/PhD/thesis.ps.gz>

CLARIT TREC-8 Manual Ad-Hoc Experiments

David A. Evans, Jeffrey Bennett, Xiang Tong, Alison Huettner, Chengxiang Zhai, Emilia Stoica

CLARITECH Corporation

Abstract. CLARITECH's submission in TREC-7 demonstrated the utility of document clustering in retrieval. We continued this work in TREC-8, using a clustered document presentation exclusively. We also added significant new functionality to the manual ad hoc user interface, integrating it with an entity extraction subsystem (upgraded and customized for TREC). Extracted entities represent an alternate set of document features. Our experiments suggest that in many cases users might construct more effective queries by moving beyond surface terms and drawing from this more abstract pool of semantic types. Despite the interface enhancements, our focus this year was on system rather than human subject performance, and we simplified the experiment design accordingly. From the users' perspective, there was only one run; the five separate submissions represent variations in post-processing. We spent minimal time preparing the initial queries. Users had 20 (instead of last year's 30) minutes for relevance judgments, and were allowed to modify the query from the start. This year, as well, we reintroduced "vector-length optimization" in the post-processing of feedback. Recent CLARITECH systems have augmented the manually generated queries with a fixed, arbitrary number of selected terms from top-ranked documents. This year, we experimented with a principled truncation of the candidate term list, and found this had a positive effect on the performance of both of our TREC-7 and TREC-8 final queries. We feel that further performance improvements are likely to be achieved only by developing several complementary techniques and applying them selectively to fine-tune individual queries. User-directed feature selection and vector-length optimization are two such promising techniques.

1 Introduction

CLARITECH's approach to manual ad hoc retrieval in TREC-7 involved the use of clustering to facilitate users' identification of relevant documents for subsequent feedback and automatic processing. Our results demonstrated the positive effect of clustering retrieved documents (vs. ordinary ranked list presentation). In particular, at all sampled time points for subjects giving relevance judgments, subjects who used clustered sets of documents out-performed those who used ranked lists. Our overall system results were quite good.

This year, we revisited the problem of clustering by adding the ability to cluster documents using a variety of document features, including entities and semantic abstractions, as well as terms. Our hypothesis was that, depending on the type of query, different document features would afford the most natural basis for organizing results. For example, questions about a specific topic might best be addressed by having retrieved results clustered primarily by entities such as person, place, organization, etc., and only secondarily by terms. In our TREC-8 experiments, we offered users the opportunity to cluster results by several such user-selected features. In addition, we shortened the amount of time that users were given to complete their reviews of documents, from a full 30 minutes per query in TREC-7 to 20 minutes per query this year. We sampled results at five-minute intervals during the 20-minute task and can also report on the relative trade-off in time on task (efficiency) vs. performance.

Subsequent to obtaining users' judgments, the CLARIT system processes the judged documents fully automatically to expand the original query and select the final set of results. Such processing depends on identifying terms in judged documents to be added to the source query vector. In the recent past, we have had good results when using a fairly large, but arbitrarily truncated set of discovered terms. This year, we returned to an approach that we used in early TREC experiments and used a principled truncation of candidate supplementary terms—a process we call "vector-length optimization". In pre-TREC-8 experiments on the TREC-7 data, using our submitted final queries from last year, we achieved more than 10% improvement over our TREC-7 results by truncating the query vector at that point where the expected contribution of an additional new term drops below a threshold of utility. Using such an approach, we achieved a higher performance on TREC-7 data with queries that averaged 50 terms vs. the 250 terms in our submitted final results. In our automatic processing of judged documents this year, we completed runs that used both our TREC-7 approach (fixed-length vectors) and our new technique (length-optimized vectors).

The two new techniques that we introduced in our work this year—(a) active use of a variety of document features, including entities, and (b) vector-length optimization—are important, general techniques for

information management, not only for information retrieval. We have used these techniques in our other track work this year; and we see in them great potential for helping to solve that very challenging problem in information processing—the fine-tuning of several complementary approaches to the individual requirements of a query or task.

2 Experiment design

For this year's TREC experiment, the 50 queries (401–450) from NIST were entered into the CLARIT system with minimal editing. We started with the text of the title, description, and narrative fields as the query, with editing by a single researcher. The researcher spent very little time on each query (well under 5 minutes), and was not permitted to retrieve any documents. Editing was limited to:

- punctuation changes (e.g., replacing commas with semicolons)
- the omission of query sentences describing non-relevant documents
- the removal of “empty” words such as *documents that discuss* or *a relevant document should include*
- repetition of nouns modified by conjoined adjectives (e.g., *genetic and environmental factors* became *genetic factors and environmental factors*)
- very occasional addition of an obviously relevant word or phrase (for example, *quilt show* in query 418)
- occasional addition of a query constraint, possibly involving extraction entities

The user's task was as always to submit the initial queries to a database consisting of the target corpora and judge the results. Results were presented as clustered groups of the top 150 documents. Users' relevance judgments were automatically collected at 5, 10, 15, and 20 minutes. Users were allowed to reformulate the initial query and retrieve potentially new results at any time during the 20-minute task. They could also terminate the task before the 20-minute time limit if they felt they had found all the relevant documents.

All the documents in the database used by the subjects were indexed with extracted entities. Extraction entities include cities, provinces, nations, personal names, employee/appointee titles, business names, and names of other organizations, such as government bodies and universities. Such entities are identified automatically using the CLARIT extraction engine, which utilizes both standard patterns (e.g., *honorific + known first name + initial + unknown word* is a standard pattern for personal names, like *Mr. Hubert M. Nar-*

malee) and large or exhaustive lists (e.g., the names of the 322 nations in the world today). Once identified, these items can be indexed as terms, but with their entity type remaining available. The set of entities in a retrieved document could be viewed by the subjects, either highlighted in context or in a separate list; the subjects could also use entities and/or entity types in constraint formulation. For example, for query 401—*What language and cultural differences impede the integration of foreign minorities in Germany?*—the user could require that all documents retrieved include the nation entity *germany*. For query 428—*What other countries besides the United States are considering or have approved women as clergy persons?*—the user could require that all documents retrieved include one or more nation entities, or that the specific nation entity *united states* be excluded. It was also possible to include constraints requiring or prohibiting a particular term or verbatim string.

The system presents the initial results as term-based document clusters, but supports clustering (and cluster summarization) by extracted entities in subsequent clustering (or reclustering) operations. All subjects had had some prior searching experience, though some were new to the CLARIT system. All subject actions, with time stamps at one-minute intervals, were written to a relational database.

The users generated a set of relevance judgments; these were further processed fully automatically to produce the submissions. Due in part to the new user interface, which simplifies the use of advanced features such as constraints, the subjects made heavy use of the constraint mechanism. Nearly three quarters of the queries contained constraints; 50% of the total had “term” constraints, and 22% had constraints involving extraction entities. Four percent were negative constraints (i.e., they excluded documents containing certain specific terms or entities).

We used the same set of user relevance judgments in four experimental runs:

- CL99SD, the “empty” run, which used neither of our new techniques
- CL99XT, in which we took advantage of extraction entities
- CL99SDopt, using vector-length optimization¹
- CL99XTopt, using both extraction entities and vector-length optimization

¹ We are omitting from this discussion our second optimized run, CL99SDopt2, in which we lowered the weights of the user-generated query to match the feedback term weights. This uniformly hurt performance. (For simplicity, we refer to CL99SDopt1 as CL99SDopt throughout this paper.)

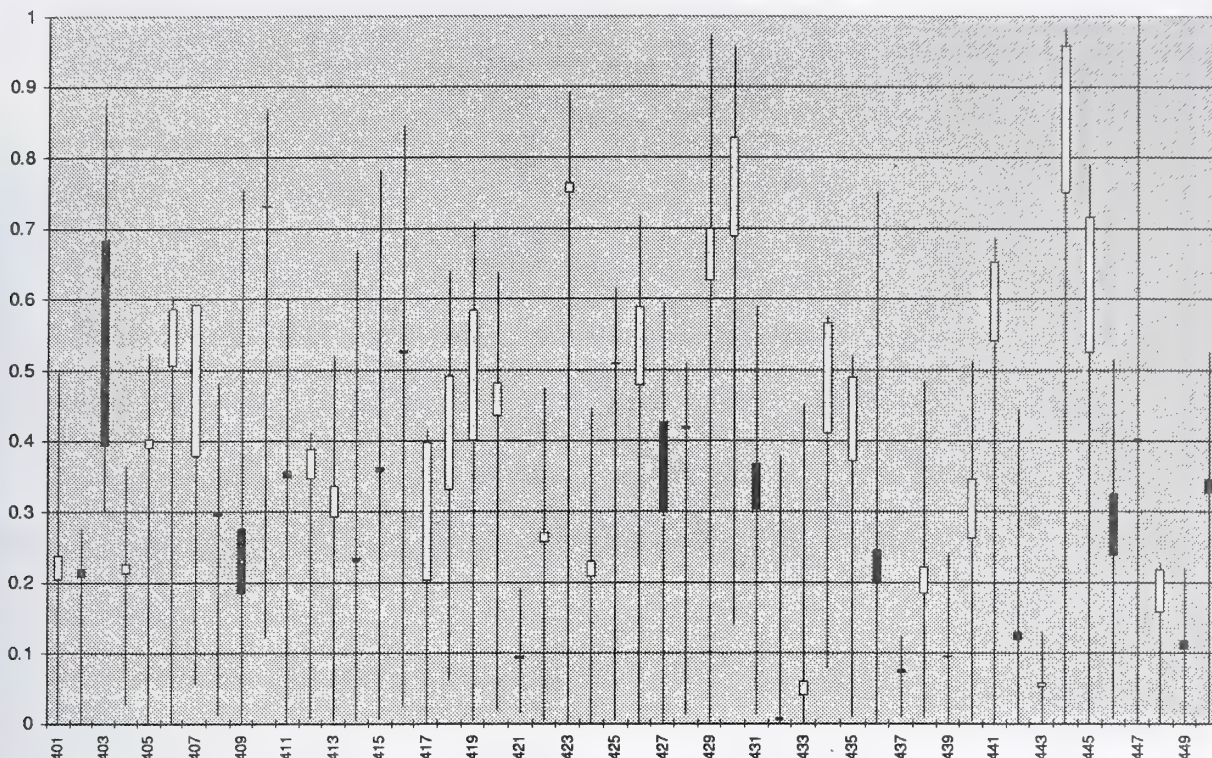


Figure 1. Comparative performance analysis: CL99XTopt vs. the group

All runs used pseudo-relevance feedback (an adaptation of the Rocchio method); the system assigned a coefficient of 0.5 to all new query terms. The system excluded documents explicitly marked non-relevant by users, and promoted marked relevant documents to the top of the ranked list. The XT runs (the two using the entity database) used constrained queries. To ensure a complete submission, these runs required two retrievals—one with and one without the constraints. All documents satisfying the constraints were returned first; if necessary, the system rounded out the top 1000 with documents from the unconstrained retrieval.

For the baseline run, CL99SD, the system removed all query constraints, and added a standard fixed-length vector of feedback terms (250). (This duplicates the approach we took in TREC-7.) CL99XT, the baseline entity run, included constraints (using the merging algorithm described above). CL99SDopt used vector-length optimization; CL99XTopt used both constraints and optimization.

The purpose of vector-length optimization is to avoid the “over-fitting” that can occur when adding too many feedback terms to a query.

Though we have observed good results in the past using a fixed-length vector of 250 terms, we often find

that reducing this number yields even better performance. In fact, reducing the vector to a mere 20 terms increases the average precision of our TREC-8 baseline run (CL99SD) from 0.3537 to 0.3638—nearly a 3% improvement.

We observe that, in general, longer documents require more feedback terms, while document sets containing many rare terms need fewer feedback terms. More specifically, there seems to be a relation between the distribution of term weights and the number of feedback terms required to maximize average precision. Sorting the candidate feedback terms by decreasing weight, the point of diminishing (and eventually negative) value occurs as the curve begins to “flatten,” as the difference in weight between successive terms approaches zero. We use a simple heuristic to estimate this point: determine the range of term weights and include all terms with weight greater than or equal to $\min + p * (\max - \min)$, where p is a parameter. We also imposed an upper limit of 250 terms on the feedback. The CL99SDopt and CL99XTopt runs described here used $p = 0.05$. (Our official CL99SDopt submission used $p = 0.1$.)²

² The two non-entity runs that we actually submitted used an older version of the CLARIT system—a version without the entity-indexing option—and $p = 0.1$. In this discussion, we ensure comparability with our XT results by substituting a new set of SD runs, using the new system and $p = 0.05$.

3 Retrieval performance

Our official results show the strong positive effect of extraction entities and the weaker positive effect of vector-length optimization.

Figure 1 details our results relative to the median. The whiskers show the entire performance range (Worst to Best); the boxes show the median and CLARITECH's (XTOpt) average precision score. If the box is white, the CLARIT score is given by the top edge, the median by the bottom edge; in these cases, we outperformed the median. If the box is black, the CLARIT score is given by the bottom edge, the median by the top edge; in these cases our performance was below median.

Table 1 shows the results for the re-run versions of the SD runs (SD and SDopt) using the same code base as for the XT runs. (Our actual submission used an older code base for the baseline runs.)

The table shows a slight improvement in average precision due to vector-length optimization (2% for SD → SDopt and 1% for XT → XTOpt). There is a much stronger effect for use of extraction entities and constraints (about 9% for SD → XT and 8% for SDopt → XTOpt). Note that the XT run used a fixed 250-term vector, while post-TREC experiments determined that a 20-term vector significantly improves the baseline performance.

The distinct clumping of values in the other columns is intriguing and suggestive of the effect of each technique (constraints and vector optimization) on the retrieval process. Initial precision and total recall

seem indifferent to entities and constraints, but respond strongly to optimization. Sustained precision seems to be aided by constraints, but actually harmed by vector optimization.

It is instructive to examine the specific types of constraints that were used, to see whether performance is sensitive to the particular form of the constraint. We have divided all constraints into four (slightly overlapping) types. General Entity constraints specify entity types—not specific entities. For instance, one such constraint might require all documents to contain a person entity. A Specific Entity constraint might require the person name *Abraham Lincoln*. A Term constraint requires the presence of a specific term that is not an entity recognized by the extraction system (e.g., *ship* or *storm*). Finally, a Negative constraint requires the *absence* of a term, entity, or entity type. Tables 2–4 reflect this analysis for four General Entity, seven Specific Entity, 25 Term, and four negative constraints. For comparison, the tables also include results for the 14 completely unconstrained queries, and the overall averages or totals.

Note that constraints were not used at all for the baseline (SD) runs, yet the average precision values nevertheless vary widely. This indicates that queries the users thought needed general entity constraints were “easy,” while those requiring negative constraints were the most difficult. Queries requiring specific entities were similarly difficult. It is also striking that actually using the constraints helps in nearly all cases.

Run	Avg. Precision	Initial Precision	Precision @ 100	Recall
CL99XTOpt	0.3765	0.9245	0.3030	3366
CL99XT	0.3730	0.9060	0.3078	3367
CL99SDopt	0.3489	0.9285	0.2732	3300
CL99SD	0.3425	0.9081	0.2766	3282

Table 1. Comparison of four CLARIT runs on standard metrics.

Average Precision	SD	XT	SDopt	XTOpt
General Entity	0.4621	0.5235	0.4659	0.5215
Specific Entity	0.2221	0.2782	0.2203	0.2723
Term	0.3186	0.3576	0.3303	0.3655
Negative	0.1908	0.2201	0.1904	0.2171
Unconstrained	0.3459	0.3457	0.3463	0.3463
All	0.3425	0.3730	0.3489	0.3765

Table 2. Average precision by constraint type, for four CLARIT runs.

Note again the striking performance variation, and the dominance of the entity runs.

Despite the general improvement due to constraints, we fear that in some cases a general constraint might dredge up large numbers of documents from the bottom of the ranked list. Since documents that satisfy the constraint are always ranked above those that do not, some relevant documents might be excluded. We will examine the effect of limiting the number of such documents in order to avoid this problem. We can imagine several more sophisticated techniques for merging constrained and unconstrained retrieved document sets.

Another parameter that merits further examination is the coefficient of feedback terms. We have traditionally assigned feedback terms lower weights than user-generated terms, yet follow-up experiments on both TREC-7 and TREC-8 have indicated that improved performance often results from assigning weights to feedback terms that are closer to the average manual term weight.

Tables 5 and 6 show the TREC-reported statistics for the runs.

Precision @ 100	SD	XT	SDopt	XTopt
General Entity	0.4100	0.4550	0.4075	0.4425
Specific Entity	0.2214	0.2871	0.2271	0.2871
Term	0.2448	0.2828	0.2400	0.2776
Negative	0.2060	0.2340	0.2020	0.2420
Unconstrained	0.2969	0.2969	0.2900	0.2900
All	0.2766	0.3078	0.2732	0.3030

Table 3. Precision at 100 documents by constraint type, for four CLARIT runs.

Average Recall	SD	XT	SDopt	XTopt
General Entity	88.5	90.3	89.0	90.0
Specific Entity	81.7	92.0	82.3	93.1
Term	57.8	59.2	57.6	58.4
Negative	65.6	69.0	65.6	69.6
Unconstrained	60.5	60.5	61.4	61.4
All	3282 (Total)	3367 (Total)	3300 (Total)	3366 (Total)

Table 4. Recall by constraint type, for four CLARIT runs.

Precision	SD	SDopt	XT	XTopt
0	0.9047	0.9163	0.9061	0.9245
0.1	0.7438	0.7509	0.7491	0.7703
0.2	0.5875	0.6030	0.5984	0.6148
0.3	0.4601	0.4704	0.4794	0.4826
0.4	0.3794	0.3927	0.4061	0.4021
0.5	0.3148	0.3320	0.3423	0.3407
0.6	0.2555	0.2665	0.2836	0.2862
0.7	0.1984	0.2165	0.2275	0.2335
0.8	0.1549	0.1699	0.1797	0.1842
0.8	0.0852	0.0924	0.1160	0.1135
1.0	0.0392	0.0439	0.0467	0.0443
Avg.Prec	0.3537	0.3682	0.3730	0.3766

Table 5. Recall level precision averages.

Docs	SD	SDopt	XT	XTopt
5	0.7600	0.8000	0.7680	0.7680
10	0.7020	0.7080	0.6920	0.6920
15	0.6453	0.6440	0.6280	0.6280
20	0.5840	0.5870	0.5730	0.5730
30	0.4860	0.4887	0.4907	0.4940
100	0.2816	0.2862	0.3078	0.3030
200	0.1969	0.2046	0.2148	0.2144
500	0.1099	0.1132	0.1173	0.1172
1000	0.0661	0.0675	0.0673	0.0673
R-Prec.	0.3709	0.3837	0.3829	0.3788

Table 6. Document level precision averages.

4 Effect of relevance judgments

In our post-TREC experiments, we compared the NIST judges' and CLARIT users' relevance judgments, and evaluated the relative impact of judgment differences on retrieval performance. Table 7 summarizes the differences between NIST and CLARIT relevance judgments for the documents that were judged by both the NIST judges and CLARIT users for the 50 topics. The agreement between the two judgments is calculated by dividing the *number with the same judgment* by the *total number of judged documents*.

Agreement is $(510 + 340) / (510 + 55 + 162 + 340)$, or 0.7966—slightly better than in TREC-7, where agreement was 0.7924 for the ranked run, 0.7717 for the clustering run, and 0.7835 for the combined run. The total number of documents judged by CLARIT users was smaller for TREC-8 (1067)³ than for TREC-7 (2216), because of the shorter time allowed for document selection.

5 Effect of timing

The cutoff for each subject's relevance feedback was 20 minutes. We reviewed the log of each session for the average number of relevant documents that had been found at each three-minute interval and graphed the result. We found that subjects tend to find a large number of documents immediately—within the first three minutes. This reconfirms our TREC-7 hypothesis that the clustered document presentation allows users to find relevant documents quickly. There is a second peak at 9 or 10 minutes, presumably a result of the first round of user feedback. Thus it seems that even 10 minutes might be a reasonable cutoff time for the relevance feedback process.

6 Conclusion

We conclude that entity extraction (with constraints) is useful for retrieval. Entity integration is an important step toward a more general information management approach involving a large variety of user-directed document features—syntactic, abstract, and semantic. The user interface for our TREC-8 experiments supported the clustering of documents based entirely on entity vectors, but this feature was rarely used. We envision a more general system in which the user could use a mixture of terms, entities, and

other more abstract types for sorting and clustering results, according to the demands of the task.

Vector length normalization is also promising, and more research is required here. We also intend to investigate the effect of feedback term weighting, and to develop more sophisticated constraint processing.

		CLARIT		<u>Total</u>
		Yes	No	
NIST	Yes	510	55	565
	No	162	340	502
<u>Total</u>		672	395	1067

Table 7. Comparison of CLARIT user judgments with NIST judgments for the same documents.

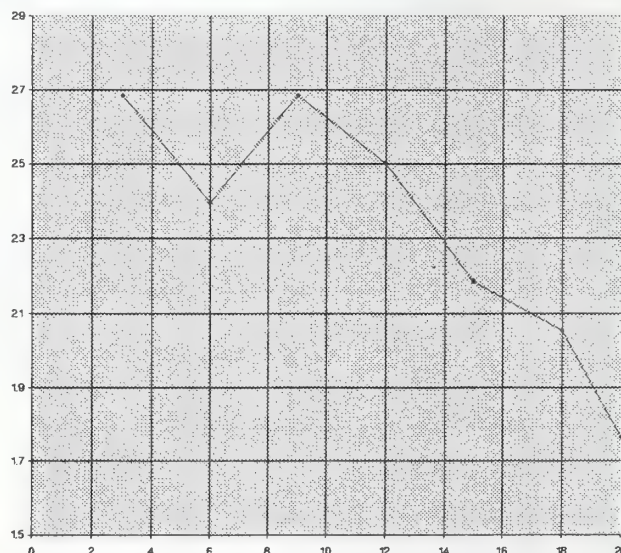


Figure 2. Numbers of documents judged (at three-minute intervals).

³ The CLARIT users' total number of judged documents was actually 1097, but 30 of the documents that our subjects judged were not judged by NIST. CLARIT users judged all of those 30 to be non-relevant, however, so there is no impact on the results.

CLARIT TREC-8 CLIR Experiments

Yan Qu, Hongming Jin, Alla N. Eilerman, Emilia Stoica, David A. Evans

CLARITECH Corporation

Abstract In the TREC-8 cross-language information retrieval (CLIR) track, we adopted the approach of using machine translation to prepare a source-language query for use in a target-language retrieval task. We empirically evaluated (1) the effect of pseudo relevance feedback on retrieval performance with two feedback vector length control methods in CLIR and (2) the effect of multilingual data merging either before or after retrieval. Our experiments show that, in general, pseudo relevance feedback significantly improves cross-language retrieval performance, and that post-retrieval merging of retrieval results can outperform pre-retrieval merging of multilingual data collections.

1 Introduction

TREC-8 marks the first occasion for CLARITECH to participate in the CLIR track. For commercial reasons, we have developed technology for English, Japanese, and Chinese CLIR. With our TREC-8 submission, we are in a position to assess how well our techniques extend to European languages.

Our approach to CLIR takes advantage of machine translation (MT) to prepare a source-language query for use in a target-language retrieval task. We developed a parameterized cross-language retrieval evaluation environment, integrating the functionality of natural language processing, retrieval, (pseudo) relevance feedback, feedback vector length optimization, MT, and data merging. For MT, we use SYSTRAN Enterprise, a commercial client-server based translation product.

Pseudo relevance feedback (PRF) has been shown, in general, to improve retrieval performance in monolingual and in cross-language retrieval using bilingual dictionaries (Ballesteros & Croft 1996). In CLIR, feedback-based query expansion can occur before query translation, after query translation, or at both places. In our pre-TREC-8 experiments, we observed that, in general, pseudo relevance feedback significantly improved retrieval performance for all the selected language pairs (English-French, English-German, and English-Italian). We calibrated our system with TREC-6 and TREC-7 CLIR topics to determine the optimal points for pseudo relevance feedback and the optimal parameter settings for the individual language pairs.

In our TREC-8 submissions, we compared two methods for controlling feedback vector length: one with a uniform number of thesaurus terms for all the topics, and the other with a varying (query-

dependent) number determined by vector length optimization.

Multilingual data merging needs to be addressed in this work because the CLIR track requires a single ranked list of retrieved documents from data collections in four languages. We distinguish pre-retrieval and post-retrieval data merging methods. Pre-retrieval data merging refers to the merging of data collections in different languages into a single multilingual data collection, while post-retrieval data merging refers to the merging of retrieval results obtained from separate data collections in different languages. Retrieval from a merged multilingual collection using multilingual topics eliminates the need for merging retrieval results, but the method can degrade the system's capability to process individual languages optimally. The post-retrieval merging method, on the other hand, allows optimization of retrieval performance for each language pair, but it requires merging of retrieval results. Our TREC-8 results show that post-retrieval merging of retrieval results can outperform pre-retrieval merging of multilingual data collections.

In the following sections, we first describe the system and the language resources employed for the TREC-8 CLIR track. Then we describe our experiments with pseudo relevance feedback and experiments in multilingual data merging, and present the evaluation results. Finally, we summarize our work.

2 System Description

We adopted MT-based query translation as our way of bridging the language gap between the source language (SL) and the target language (TL).

We implemented three methods of pseudo relevance feedback (PRF) for bilingual retrieval. The simple MT-based query translation and the PRF methods are illustrated in Figure 1. Figure 1(a) illustrates query translation without expansion. In this configuration, the topics in a source language are translated using the MT engine into texts in the designated target language, which are then used for retrieval from a target language database. Figure 1(b) illustrates query expansion prior to translation. Here each topic in a source language (SL) is first augmented with N thesaurus terms extracted from the top M subdocuments retrieved from a SL database. The top M subdocuments are assumed to be relevant to the query. The resulting topic, which consists of the original query text and the additional thesaurus terms in SL, is then sent to the MT engine. The translation of the source language query text

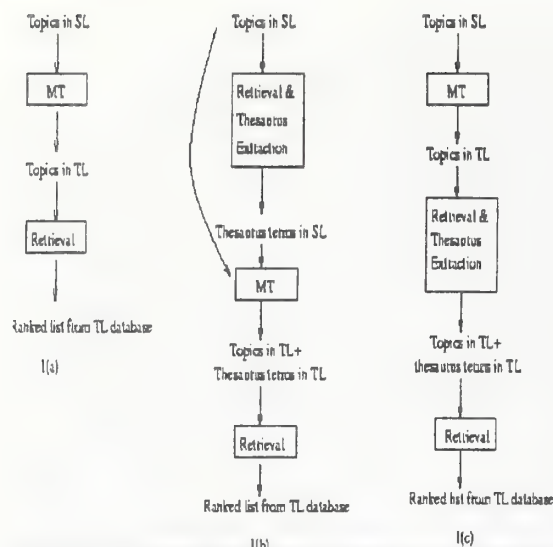


Figure 1. CLIR with MT-based query translation and pseudo relevance feedback

and the thesaurus terms is used for retrieval from a target language database. In post-translation query expansion (Figure 1(c)), the original query text is first translated via the MT engine. Then the translated query text is augmented using the feedback process. The resulting topic, which consists of the translated query topic and the thesaurus terms in the TL, is then used for retrieval from a TL database. The combined feedback method unites the feedback process prior to translation in Figure 1(b) and the feedback process after translation in Figure 1(c). For details on the CLARIT term extraction methods and the retrieval engine, the reader is referred to (Milic-Frayling et al. 1998).

For CLIR involving more than two languages, we decompose the task into bilingual retrieval from the source language to the individual target languages, then merge the retrieval results.

3 Linguistic Resources

For processing the English corpus and queries, we used the CLARIT English NLP module, which consists of a parser and a morphological analyzer that utilize the English lexicon and grammar to identify linguistic structures in texts (Milic-Frayling et al. 1998). The CLARIT NLP module supports discovery of various types of linguistic structures, such as simplex and complex noun phrases (NPs), verbs, and other selected constituents.

The English grammar was adapted for use in German, French, and Italian NLP. Necessary modifications were made to accommodate specific categories of each language.

For German NLP, we automatically extracted a core lexicon from the German lexicon distributed by the Linguistic Data Consortium (LDC). The resulting lexicon, with 318,809 entries, specifies word surface forms, their parts of speech, and normal forms.

For French and Italian NLP, we manually developed lexicons of closed-class categories that are sufficient to achieve mostly correct phrase segmentation. In addition, punctuation marks and special symbols, as found in multilingual texts, were collected and used to supplement the core lexicons. No morphological normalization was done for either language, even though a design for the French normalization had been completed.

For all four languages, we also manually constructed lexicons of stop words, which included extraneous words and their inflected forms (e.g., *document, relevant, report* in English; *document, pertinent, rapport, rapporteur* in French; *Dokument, relevant, Bericht* in German; and *documento, rilevante, rapporto* in Italian). The stop words were selected from the TREC-6 and TREC-7 topics.

For all the experiments reported in this paper, we indexed the data collections of individual languages using simplex NPs and all attested sub-terms. The English topics and their French/German/French translations were processed similarly into simplex NPs and decomposed into all attested sub-terms.

We used the SYSTRAN Enterprise software for translating the queries. The client-server configuration of this software allows us to integrate SYSTRAN's translation capability into our evaluation environment by calling the client API. The client API takes as input the source language query (plus feedback terms if feedback is used) stored in a file and the specific language pair for translation, and returns a file with the translation of the source text to the application program. Query translation is a black box process to the application program. The language pairs selected for the TREC experiments included English-French, English-German, and English-Italian.

4 Pre-TREC System Calibration

In preparation for the TREC-8 CLIR track, we performed experiments to calibrate the components of the CLIR evaluation environment. We focused on testing the effectiveness of pseudo relevance feedback on the English monolingual retrieval and the English-to-French/German/Italian bilingual retrieval. Two feedback-vector length control methods were tuned: one with uniform vector length, i.e., the same number of feedback terms for all topics, and the other with varying vector length optimized for individual topics. We used Rocchio as the thesaurus term extraction method, as it was observed to generate the greatest improvement of retrieval performance in our previous monolingual experiments. We conducted experimental runs over TREC-6 and TREC-7 CLIR topics to obtain: (1) the optimal place for pseudo relevance feedback, and (2) the optimal parameter settings for the two feedback vector length control methods.

4.1 Feedback with uniform-length vectors

For PRF using uniform-length vectors, we focused on two parameters: (1) N_d , the number of subdocuments selected for thesaurus extraction, and (2) N_t , the number of terms extracted from the set of subdocuments for augmenting the original query vector. For bilingual retrieval, we also tested the optimal place (pre-translation, post-translation, and combined positions) or pseudo relevance feedback should apply. The experiments were conducted using English as the query language. For the English monolingual retrieval and the English-to-French/German bilingual retrieval, we used TREC-6 English topics, and evaluated the results using relevance judgments for TREC-6 topics. For the English-to-Italian bilingual retrieval, we used TREC-7 English topics and TREC-7 relevance judgments.¹

In the English-to-French/German/Italian bilingual retrieval, we observed that, for all the language pairs, all three pseudo relevance feedback methods significantly improved Average Precision and Recall, compared to their respective no feedback (NF) baseline runs. In particular, post-translation query expansion yielded the greatest improvement in both average precision and recall for all the language pairs. The optimal settings obtained from the calibration are $N_d = 50$ subdocuments and $N_t = 75$ terms for English monolingual, and $N_d = 25$ subdocuments and $N_t = 50$ terms for English-to-French/German/Italian, respectively.

4.2 Feedback with optimized-length vectors

The uniform-length vector method adds the same number of terms to each profile. In contrast to this, the optimized-length vector method dynamically computes the number of terms to be added for query expansion, using the curve of terms' weights. The algorithm was developed based on the observation that there seems to be a correlation between the change in slope of the curve of the terms' weights and the average precision of a query.

The algorithm uses the first N weights (arranged from highest to lowest weights), and adds a term for query expansion if its weight satisfies the following condition:

$$w(t) \geq \min + \text{perc} * (\max - \min)$$

where \min is the smallest weight, \max is the largest weight, and perc is a constant. The method aims to provide the maximum benefit from feedback, while reducing the number of terms required for feedback.

For the optimized-length vector method, we tuned two parameters: (1) N_m , the maximum number of terms extracted from a set of subdocuments, for

which we experimented with N as 80 and 250, and (2) perc , for which we tried the values 0.25, 0.1, 0.05, and 0.01. For bilingual retrieval experiments, we selected the top 25 subdocuments to be used for term extraction and the post-translation feedback method, as they were observed to give the best retrieval performance in general for the uniform-length vector method. The training experiments were conducted using English as the query language. We used the same set of topics and data collections as described in section 4.1. Compared with the experiments with no feedback (NF), PRF using the optimized-length vector method also demonstrated significant improvements in Average Precision and Recall. The optimal settings obtained from the calibration are $\text{perc} = 0.25$ and $N_m = 250$ terms for English monolingual, and $\text{perc} = 0.05$ and $N_m = 80$ terms for English-to-French/German/Italian, respectively.

5 TREC-8 Experiments

All of our CLIR submissions used automatic query processing, with English as the topic (source) language and with the combined fields of title, description, and narrative as the body of the query.

5.1 Experiments using Pseudo Relevance Feedback

To evaluate the effectiveness of the two vector control approaches in CLIR, we conducted a baseline run (*CLARITrmnf*) by first obtaining the French, German, and Italian translations of the source English topics, and then performing monolingual retrieval for the four languages from their respective databases without using any feedback mechanism. Then we combined the four retrieved result lists into a combined result list using their raw similarity scores.

We submitted two runs, *CLARITrmwf1* (PRF with uniform-length vectors) and *CLARITrmwf3* (PRF with optimized-length vectors), to compare the effectiveness of vector length optimization. With both runs, we first indexed each data collection individually, and obtained a ranked list from each collection. The result lists were then merged based on raw similarity scores. The two runs were conducted using the PRF settings specified in sections 4.1 and 4.2.

Our experiments in Table 1 demonstrated that, in general, PRF using both vector length control methods improved retrieval performance. In particular, both methods yielded significant improvement in Recall, Average Precision, Exact Precision, and Precision at 100 documents. Only Initial Precision decreased. The optimized-length vector method outperformed the uniform-length vector method in Average Precision, Initial Precision, and Exact Precision, but underperformed the uniform-length method in Recall and Precision at 100 documents. Such results are consistent with our observations with TREC-7 topics. Vector-length

¹ However, since the relevance judgments for TREC-7 topics were made based on the combined result list rather than results for individual languages, we treated the Italian results only as suggestive.

optimization seems to be a promising technique, but requires more research into its effectiveness.

5.2 Experiments on Data Merging

We evaluated three multilingual data merging methods to obtain a single ranked list for the purpose of TREC-8 CLIR track submission.

The first experiment (*CLARITdmwf*) used pre-retrieval data merging, i.e., we merged collections of English, French, German, and Italian documents into a single multilingual data collection, and indexed the multilingual collection. The topics were translated from the source language to the target languages and were merged together to form multilingual topics. Retrieval was done using the multilingual topics to obtain a single result list from the multilingual data collection. Pseudo relevance feedback was conducted for obtaining the optimal retrieval performance. For text processing, we used a combined lexicon consisting of all the lexicons for four languages and an adapted version of the English grammar. This run was designed as a baseline to be compared with two runs using post-retrieval result merging (*CLARITrmwf1* and *CLARITrmwf2*).

In *CLARITrmwf2*, we used normalized similarity scores rather than raw similarity scores as in *CLARITrmwf1*. First, we indexed each collection individually and obtained a ranked list from each collection. Then we reconstructed new databases using the N documents from each ranked list (in TREC, $N = 1000$) and re-computed the similarity scores for each new database. We then merged ranked lists into a single combined ranked list based on the recomputed similarity scores. Table 2 presents the retrieval performance statistics for the three runs.

The performance statistics demonstrate that post-retrieval merging of retrieval results can outperform pre-retrieval merging of data collections. Specifically, post-retrieval merging significantly improved Recall, Average Precision (except in *CLARITrmwf2*), and Exact Precision. Initial Precision was decreased for both set of topics.

The experimental results for TREC-8 topics are consistent with our observations with TREC-7 topics: merging with score normalization underperformed merging using raw similarity scores. One possible reason is that the new databases for score re-computation are too small (i.e., $N = 1000$ documents) for the similarity scores to be reliable. Another possible reason is that in the CLARIT system, the *idf* scores are computed using subdocuments. If the document lengths vary greatly across databases, the number of subdocuments used for *idf* computation will vary greatly even when the number of documents selected is uniform across databases. We intend to do further research on this issue in our future work.

6 Summary

Our TREC-8 experiments demonstrated that pseudo relevance feedback can be used to improve retrieval performance significantly in MT-based CLIR. The feedback vector length optimization method yields promising results, but requires more research into its effectiveness.

Post-retrieval result merging allows the optimization of retrieval performance for each language pair and has been demonstrated to outperform the pre-retrieval data merging method. However, effective techniques for score normalization for result merging require further investigation.

References

- [Ballesteros & Croft, 1996] Ballesteros, L., and Croft, W.B. 1996. Dictionary methods for cross-lingual information retrieval. In *Proceedings of the 7th International DEXA Conference on Database and Expert Systems*, 791—801.
- [Milic-Frayling et al. 1998] Milic-Frayling, N.; Zhai, C.; Tong, X.; Jansen, P.; and Evans, D. A. 1998. Experiments in query optimization, the CLARIT system TREC-6 report. In *Proceedings of the 6th Text REtrieval Conference (TREC-6)*, 415—454.

Run	Recall	Avg. Precision	Initial Precision	Exact Precision	Prec. 100 docs
1. CLARITrmwf1 (incr./Decr. Over (3))	1807 (13.5%)	0.2297 (25.0%)	0.6198 (-10.9%)	0.2717 (24.0%)	0.2661 (23.5%)
2. CLARITrmwf3 (incr./Decr. Over (3)) (incr./Decr. Over (1))	1789 (12.4%) (-1.0%)	0.2357 (28.3%) (2.6%)	0.6865 (-1.3%) (10.8%)	0.2809 (28.2%) (3.4%)	0.2475 (14.9%) (-7.0%)
3. CLARITrmnf (unofficial, baseline)	1592	0.1837	0.6953	0.2191	0.2154

Table 1: Performance statistics for CLARITrmwf1, CLARITrmwf3, and CLARITrmnf

Run	Recall	Avg. Precision	Initial Precision	Exact Precision	Prec. 100 docs
1. CLARITrmwf1 (incr./Decr. Over (3))	1807 (25.8%)	0.2297 (8.0%)	0.6198 (-7.9%)	0.2717 (9.9%)	0.2661 (17.9%)
2. CLARITrmwf2 (incr./Decr. Over (3)) (incr./Decr. Over (1))	1626 (13.2%) (-10.0%)	0.2036 (-4.3%) (-11.4%)	0.6032 (-10.3%) (-2.7%)	0.2514 (1.7%) (-7.5%)	0.2429 (7.6%) (-8.7%)
3. CLARITdmwf (baseline)	1436	0.2127	0.6726	0.2473	0.2257

Table 2: Performance statistics for CLARITrmwf1, CLARITrmwf2, and CLARITdmwf

CLARIT TREC-8 Experiments in Searching Web Data

Jeffrey Bennett, Xiang Tong, David A. Evans

CLARITECH Corporation

Abstract CLARITECH submitted two baseline content-only runs and completed two additional content+link runs in the TREC-8 Web Track. These represent our first serious attempt to deal with Web data, and our first automatic runs in several years. The first question was whether CLARIT would perform as well on Web data as on more traditional text. We found that, with extensive pre-processing of the raw data prior to indexing, the automatic retrieval system actually performed better on Web data than on Ad Hoc data. For the link runs, we implemented a version of the HITS algorithm [Kleinberg 1997], originally developed at IBM. Our version optimized HITS for the CLARIT environment, but also reflected some constraints imposed by limited resources. Unable to develop and sufficiently test our own matrix-processing library in time, we used a commercial product for the number crunching. Performance on the link runs was poor, but failure analysis suggests many ways to improve it.

1 Introduction

Even casual inspection of Web data reveals how different it is from traditional newswire or article text. Most obviously, it contains extensive HTML mark-up. Even apparent plain text may conceal many types of meta and tag data, as the example in Figure 1 shows.

```
<head>
<META name="keywords" content="waste water, biosolids,
waste treatment, geneva, marsh creek">
<title>Marsh Creek Waste Water Treatment Plant</title>
</head>
<body bgcolor="#ffff99" text="#006633" link="#993300"
vlink="#CC0099">
<center><a name="top"></a></center>
<br clear="all">

<p></p><br><p></p><br>
<h1> Marsh Creek</h1>
<h1><a name="Marsh Creek Treatment Plant"> Waste
Water Treatment Plant
</a></h1>
```

Figure 1. Example of web text with HTML markup

Clearly, something must be done; a naïve parser, attempting to find words in the above text, might extract terms like "Creek</h1>" "Plant", and "<title>Marsh"—words unlikely to be found in any lexicon.

To address this problem, the CLARITECH web system does extensive pre-processing to "sanitize" the text, while preserving important information encoded in the mark-up. Non-semantic tags, such as "
", "<body bgcolor>", or "<center>", are simply discarded; the system processes certain other tags with greater care.

Specifically, it removes keywords, header data (e.g., the site URL, server address, last-modified date, document length) and hyperlinks from the main text, and stores them all in separate fields. (The link field retains an offset into the main text for future context recovery.) The system also records forms, images, and an image count for possible future use. This pre-processing step allows CLARIT to work on the plain text and deliver quite satisfactory performance on the content-only runs, particularly on recall. Our best run was at or above median recall on 88% of the queries. Precision was less impressive, but still slightly above median. Surprisingly, the performance of the automatic system on Web data was better than the same system's performance on the ad hoc corpora. We performed a large number of ad hoc automatic runs (though we made no submissions to this track), and found that our system performed at or slightly above median for TREC-8 queries. This is encouraging, since we have made no attempt to "tune" our system for automatic retrieval.

Having essentially no time for pre-experiment evaluations, we took a very bold approach to the content + link runs. After a brief literature review, we decided to implement a "CLARITized" version of the Hyperlink-Induced Topic Search (HITS) algorithm. The idea is that hyperlinks can be viewed as implicit annotations that encode human judgments about relevance. Confronted with the thousands or even tens of thousands of "relevant" documents that might be returned by a general query, the link information can be used to extract a much smaller number of "authoritative" sources on the topic. These pages are likely to be of greater utility to the user than a "ranked" list of hundreds of nearly indistinguishable documents.

A page linked to by many relevant pages is said to have "authority," even when the page itself is not high-scoring. Conversely, pages containing links to many relevant documents are called "hubs." As with authorities, the hub pages themselves may be low-scoring; they may in fact have little or no content beyond the list of hyperlinks. The premise of the HITS algorithm is that authorities and hubs have more utility than isolated relevant documents, and should therefore be ranked higher.

As a first step, we used the connectivity information supplied by TREC (rather than examining our own link fields in context), and only considered links involving the top 250 or so documents in the ranked list. We used a brute-force method for the link matrix calculations, and then simply front-loaded the top 30 authorities. Our approach was "bold" in that it did not examine the content of the authority pages, and was biased toward densely linked pages that conventional retrieval had missed. (On average, 22 of the 30 authorities came from below the top 1,000.)

2 Experiment design

The content-only runs used standard CLARIT retrieval with pseudo-relevance feedback. We generated the queries automatically using the "title" and "description" fields; assigning a higher weight to title terms. The two runs we submitted differ only in the parameters used for feedback: CL99WebM extracted terms from high-scoring subdocuments in the top 10 documents, and added up to 30 new terms; CL99WebH examined the top 30 documents and added up to 50 terms. The system weighted new terms lower than existing query terms. We submitted the top 1,000 documents returned by a second retrieval using the augmented queries.

The content + link runs started with the output of the baseline runs, then analyzed the hyperlink information in top-ranked documents to identify the top 30 authority and hub pages. To produce the link submissions, we promoted these pages to the top of the ranked lists. We tested the utility of the HITS algorithm in our environment, as well as the effect of some slight modifications we made in order to capitalize on the strengths of CLARIT.

We begin by constructing a base set of N pages (S), and limiting our search for authoritative pages to this set. N is arbitrary; we chose 1,000 as a practical limit (imposed mostly by limited time and computing resources). Using the connectivity information provided by TREC, we expanded the initial result list according to the following algorithm: number the pages $\{1, 2, \dots, n\}$, and iterate through them in rank

order. Copy the i th page to S , then also copy all the documents that the i th page links to as well (if they are not already in the S). Proceed until S contains 1,000 unique documents. The average number of links per page in the collection was 6.1; accordingly, the average "depth" of the expansion step was around 230 to 250 documents. We felt this would bias the results toward documents linked to the most highly-ranked pages; expanding further might have turned up densely linked networks of low-ranked, non-relevant documents. Also, we were unable to process large (3,000—5,000 square) matrices using the brute-force methods we employed. The actual depth varied considerably, from a minimum of 41 to a maximum of 410 documents. (Another approach, which we did not explore, would be to mine deeper by examining the links and adding them selectively, rather than simply dumping them all into S .)

We then constructed an $n \times n$ "adjacency matrix" A , whose (i,j) th entry is set equal to a non-zero value if page i links to page j , and zero otherwise. The simplest approach would be to set link entries to 1. Our bold approach biased the results toward unretrieved documents by computing CLARIT-term-based similarity between "best-hit" subdocuments when both pages appeared in the top 1,000, and 1 otherwise. Many of these similarity scores were near zero, effectively discounting links between dissimilar documents among the top 1,000.

According to the HITS algorithm [Chakrabarti et al. 1999], we assign authority and hub weights to each document in S . Let the authority score for the i th document $x_i = \sum_{j \rightarrow i} y_j$; the sum of all pages j that link

to i . Similarly, compute the i th hub score $y_i = \sum_{i \rightarrow j} x_j$;

the sum all pages i links to. There is a natural feedback effect here, since authority and hub pages are closely related: good hubs are good because they point to many authorities; and, authorities are authoritative precisely because they are being linked to by good hubs. Given the contents of the adjacency matrix, and considering the set of hub and authority scores as vectors, we can derive a process that expresses this mutually reinforcing relationship and reduces it to a standard operation in linear algebra. By filling the matrix with similarity scores between 0 and 1 (non-negative values), we guarantee that the matrix processing will converge on pages containing the most dense linkage patterns.

Specifically, the authority vector update formula can be expressed as $x \leftarrow (A^T A)x$; similarly, the hub

vector update function is $y \leftarrow (AA^T)y$. This is equivalent to performing *power iterations* on $A^T A$ and AA^T ; such iteration (given non-negative coefficients) converges on the principal eigenvectors of the associated matrices.

To find the most authoritative sources, we generated both matrices, used a commercial package to calculate their principal eigenvectors, and sorted the resulting vectors by decreasing weight. We now had ranked authority and hub vectors for the documents in S . The final step was to merge the two vectors to obtain a single ranked list of "authorities." We did the merge by computing a total score for each document. If document D1 was ranked first in the authority vector and third in the hub vector, it received a score of $1/1 + 1/3 = 1.333$. If D2 was ranked second in both vectors, it received a score of $1/2 + 1/2 = 1$, and so on for all documents. We moved the top N authorities to the top of the original ranked list, where N was arbitrarily (based on a literature review) set to 30, and that was the submission.

3 Retrieval performance

The results show reasonable performance for the content-only runs, and poor performance for the link runs (see Table 1.)

Since our link submissions were not judged, and the HITS algorithm may well find documents that are non-relevant by TREC standards (i.e., collections of links without content), we expected the link run scores to be low, even if our approach was working and returning helpful documents.

Noting that most of the documents in the top 30 were originally ranked below 1,000, we thought that perhaps our approach had been a little too bold. A more conservative algorithm might perform a simple resorting of the original results, without bringing in previously unretrieved documents. We did two follow-up runs using this more timid algorithm; see Table 2 for these results.

These runs fall precisely in the middle; the most irrelevant documents have been discarded, improving all performance measures, but results still fall far short of the content-only runs. See Figure 1 for the P-R curve.

4 Failure analysis of the link runs

Upon closer analysis, these factors were not sufficient to explain the poor performance. In fact, many pages had links to common "web statistics" and "hit counter" sites. There were also many links to pages

giving mutual fund indices and commercial ad sites. Since we did not examine the pages for relevance, and actually preferred pages that were not returned by the conventional search, such irrelevant links formed the densest patterns!

One such site appeared in the top 30 on seven different queries, 11 were referenced by six queries, and 22 were referenced by three queries. Overall, 13% of the top-ranked documents were non-unique. If the queries are independent, all the top-ranked documents should be unique, so this degree of overlap is a sure indication something is wrong.

Our analysis confirms at least that we know the linkage pattern detection is working properly. In future work we could try several different approaches to address this problem. We could generate a stop-list of known statistics, counter, and commercial sites. We could "sanity check" the final results for overlap of top-ranked documents. We could impose a minimum score or other relevance test on documents that were not initially returned (or perhaps we should limit ourselves to resorting documents in the top 1,000, and not look beyond the initial results at all). Another strategy might be to weight the links using an adapted IDF formula. We are looking for "discriminating" links—document sets that link to each other but not to lots of other documents scattered throughout the database. A counter site might be linked to by hundreds of unrelated documents in the database; it would be assigned a very low "IDF" score, and assigned a low value in the matrix (or discarded entirely). We might even use clustering to try to identify related groups of documents within a link network.

5 Conclusion

We were encouraged by our relatively good performance on the content-only runs, particularly since our system has not been optimized to work without user feedback. The link runs were disappointing, but we can see why, and we have many ideas to address the problems. In keeping with our general belief that the next breakthrough in performance will come from customizing an approach for each query from a number of complementary techniques, we will explore the range of conditions for which link analysis is appropriate. It would appear to be most applicable for sorting through large sets of nearly equally high-scoring documents, as would result from very general queries. When the query is more specific, perhaps traditional CLARIT processing is enough; link analysis might tend to decrease performance in such cases. In the TREC-8 manual Ad Hoc task, we discovered that users added boolean

constraints to nearly 75% of the queries; this implies that the queries were generally quite specific this year. This may have contributed to the poor performance, especially of the CMLnk and CHLnk runs.

Finally, we recognize that this technique is actually quite general, and could be applied to non-Web data. We can imagine generalizing the concept of a "link" to mean, for instance, references to the same RDB field across multiple databases, similar subdocuments or themes across different documents, detected "events" in chronological newswire databases, etc. The concept has already been applied to citations in databases of academic papers.

References

[Kleinberg 1997] Kleinberg, J.M. "Authoritative sources in a hyperlinked environment." *In Proceedings of Ninth ACM-SIAM Symposium on Discrete Algorithms*, 1998, and IBM Research Report RJ 10076, May 1997.

[Chakrabarti et al. 1999] Chakrabarti, S., Jon Kleinberg, et al. *Mining the Link Structure of the World Wide Web*. February 1999.

Run	Average Precision	Initial Precision	Precision @ 100	Recall
CL99WebM	0.2885	0.5431	0.1724	1924
CL99WebH	0.2838	0.5315	0.1806	1933
CL99WebMLnk	0.1237	0.2321	0.1518	1923
CL99WebHLnk	0.1266	0.2501	0.1538	1929

Table 1. Comparison of all runs

Run	Average Precision	Initial Precision	Precision @ 100	Recall
CMLnk	0.2043	0.3767	0.1728	1924
CHLnk	0.2055	0.3967	0.1782	1933

Table 2. Comparison of "conservative" link runs

P-R Curves

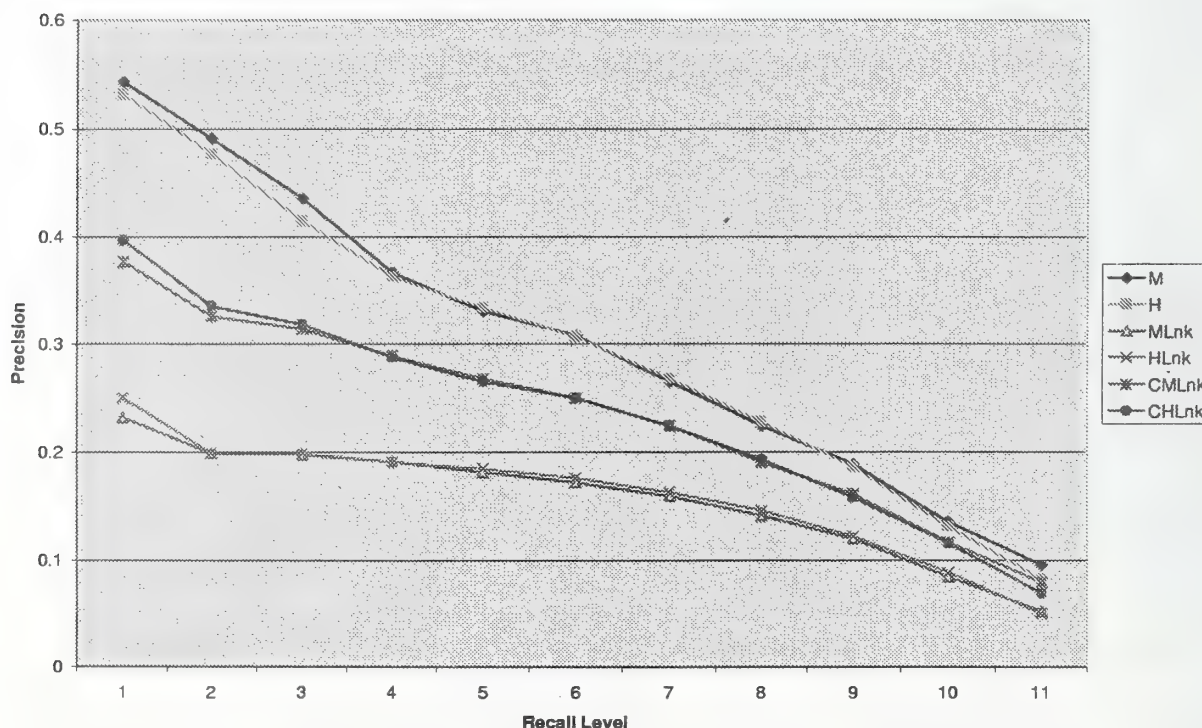


Figure 2. P-R curves for all web runs

Question-Answering Using Semantic Relation Triples

Kenneth C. Litkowski
CL Research
9208 Gue Road
Damascus, MD 20872
ken@clres.com
<http://www.clres.com>

Abstract

This paper describes the development of a prototype system to answer questions by selecting sentences from the documents in which the answers occur. After parsing each sentence in these documents, databases are constructed by extracting relational triples from the parse output. The triples consist of discourse entities, semantic relations, and the governing words to which the entities are bound in the sentence. Database triples are also generated for the questions. Question-answering consists of matching the question database records with the records for the documents.

The prototype system was developed specifically to respond to the TREC-8 Q&A track, with an existing parser and some existing capability for analyzing parse output. The system was designed to investigate the viability of using structural information about the sentences in a document to answer questions. The CL Research system achieved an overall score of 0.281 (i.e., on average, providing a sentence containing a correct answer as the fourth selection). The score demonstrates the viability of the approach. Post-hoc analysis suggests that this score understates the performance of the prototype and estimates that a more accurate score is approximately 0.482. This analysis also suggests several further improvements and the potential for investigating other avenues that make use of semantic networks and computational lexicology.

1. Introduction

CL Research is primarily focused on investigating the manner in which computational lexicons can be used for natural language processing tasks. This research primarily involves the development of methods for constructing computational lexicons (particularly through analysis of machine-readable dictionaries) and examining ways that these lexicons

can be used in such tasks as word-sense disambiguation and text summarization.

The CL Research question-answering prototype extended functionality of the DIMAP dictionary creation and maintenance software, which includes some components intended for use as a lexicographer's workstation.¹ The TREC-8 Q&A track provided an opportunity not only for examining use of computational lexicons, but also for their generation as well, since many dictionaries (particularly specialized one) contain encyclopedic information as well as the usual genus-differentiae definitions. The techniques developed for TREC and described herein are now being used for parsing dictionary definitions to help construct computational lexicons that contain more information about semantic relations, which in turn will be useful for natural language processing tasks, including question-answering.

2. Problem Description

Participants in the TREC-8 QA track were provided with 200 unseen questions to be answered from the TREC CD-ROMs, (about 1 gigabyte of compressed data), containing documents from the *Foreign Broadcast Information Service*, *Los Angeles Times*, *Financial Times*, *Congressional Record*, and *Federal Register*. These documents were stored with SGML formatting tags. Participants were given the option of using their own search engine or of using the results of a "generic" search engine. CL Research chose the latter, obtaining 200 megabytes of data, with the top 200 documents retrieved by the search engine. These top documents were provided a couple of weeks before the deadline.

¹Demonstration and experimental versions of DIMAP are available at <http://www.clres.com>.

Participants were then required to answer the 200 questions in either 50-byte answers or by providing a sentence or 250-byte string in which the answer was embedded. For each question, participants were to provide 5 answers, with a score attached to each for use in evaluating ties. NIST evaluators then judged whether each answer contained a correct answer. Scores were assigned as the inverse rank. If question q contained a correct answer in rank r , the score received for that answer was $1/r$. If none of the 5 submissions contained a correct answer, the score received was 0. The final score was then computed as the average score over the entire set of questions.

In the prototype implementation, CL Research submitted sentences, although for some types of questions, answers were also developed for potential 50-byte submission.

3. System Description

The CL Research prototype system consists of four major components: (1) a sentence splitter that separated the source documents into individual sentences; (2) a parser which took each sentence and parsed it, resulting in a parse tree containing the constituents of the sentence; (3) a parse tree analyzer that identified important elements of the sentence and created semantic relation triples stored in a database; and (4) a question-answering program that (a) parsed the question into the same structure for the documents, except with an unbound variable, and (b) matched the question database records with the document database to answer the question. The matching process first identified candidate sentences from the database, developed a score for each sentence, and chose the top 5 sentences for submission.

3.1 Sentence Identification in Documents

The parser (described more fully in the next section) contains a function to recognize sentence breaks. However, the source documents do not contain crisply drawn paragraphs that could be submitted to this function. Thus, a sentence could be split across several lines in the source document, perhaps with intervening blank lines and SGML formatting codes. As a result, it was first necessary to reconstruct the sentences, interleaving the parser sentence recognizer.

At this stage, we also extracted the document identifier and the document date. Other SGML-tagged fields were not used. The question number, document

number, and sentence number provided the unique identifier when questions were answered to extract the appropriate sentence from the document.

For the TREC-8 QA run submitted to NIST, only the top 10 documents (as ranked by the search engine) were analyzed. Overall, this resulted in processing 1977 documents from which 63,118 sentences were identified and presented to the parser. Thus, we used an average of 31.9 sentences per document or 315.5 sentences in attempting to answer each question.

3.2 Parser

The parser used in TREC-8 (provided by Proximity Technology) is a prototype for a grammar checker. The parser uses a context-sensitive, augmented transition network grammar of 350 rules, each consisting of a start state, a condition to be satisfied (either a non-terminal or a lexical category), and an end state. Satisfying a condition may result in an annotation (such as number and case) being added to the growing parse tree. Nodes (and possibly further annotations, such as potential attachment points for prepositional phrases) are added to the parse tree when reaching some end states. The parser is accompanied by an extensible dictionary containing the parts of speech (and frequently other information) associated with each lexical entry. The dictionary information allows for the recognition of phrases (as single entities) and uses 36 different verb government patterns to create dynamic parsing goals and to recognize particles and idioms associated with the verbs (the context-sensitive portion of the parser).

The parser output consists of bracketed parse trees, with leaf nodes describing the part of speech and lexical entry for each sentence word. Annotations, such as number and tense information, may be included at any node. The parser does not always produce a correct parse, but is very robust since the parse tree is constructed bottom-up from the leaf nodes, making it possible to examine the local context of a word even when the parse is incorrect. In TREC-8, the parse output was unusable for only 526 of the 63,118 sentences (0.8 percent). Usable output was available despite the fact that there was at least one word unknown to the parsing dictionary in 5,027 sentences (8.0 percent).

3.3 Document and Question Database Development

The key step in the CL Research question-answering prototype was the analysis of the parse trees to extract semantic relation triples and populate the databases used to answer the question. A **semantic relation triple** consists of a discourse entity, a semantic relation which characterizes the entity's role in the sentence, and a governing word to which the entity stands in the semantic relation.

In general terms, the CL Research system is intended to be part of a larger discourse analysis processing system (Litkowski & Harris, 1997). The most significant part of this system is a lexical cohesion module intended to explore the observation that, even within short texts of 2 or 3 sentences, the words induce a reduced ontology (i.e., a circumscribed portion of a semantic network such as WordNet (Fellbaum, 1998) or MindNet (Richardson, 1997)). The objective is to tie together the elements of a discourse (in this case, a document) using lexical chains and coreference to create a hierarchical characterization of a document. The implementation in TREC-8 does not attain this objective, but does provide insights for further development of a lexical cohesion module.

The first step of this discourse processing is the identification of suitable discourse entities. For TREC-8, this involved analyzing the parse tree **node** to extract numbers, adjective sequences, possessives, leading noun sequences, ordinals, time phrases, predicative adjective phrases, conjuncts, and noun constituents as discourse entities. To a large extent, these entities include, as subsets, named entities and time expressions as single entities (although not specifically identified as such in the databases).

The semantic relations in which entities participate are intended to capture the semantic roles of the entities, as generally understood in linguistics. This includes such roles as agent, theme, location, manner, modifier, purpose, and time. For TREC-8, we did not fully characterize the entities in these terms, but generally used surrogate place holders. These included "SUBJ," "OBJ," "TIME," "NUM," "ADJMOD," and the prepositions heading prepositional phrases.

The governing word was generally the word in the sentence that the discourse entity stood in relation to. For "SUBJ," "OBJ," and "TIME," this was generally the main verb of the sentence. For prepositions, the governing word was generally the noun or verb that

the prepositional phrase modified. (Because of the context-sensitive dynamic parsing goals that were added when a verb or a governing noun was recognized, it was possible to identify what was modified.) For the adjectives and numbers, the governing word was generally the noun that was modified.

The semantic relation and the governing word were not identified for all discourse entities, but a record for each entity was still added to the database for the sentence. Overall, 467,889 semantic relation triples were created in parsing the 63,118 sentences, an average of 7.4 triples per sentence.

The same functionality was used to create database records for the 200 questions. The same parse tree analysis was performed to create a set of records for each question. The only difference is that one semantic relation triple for the question contained an unbound variable as a discourse entity. The question database contained 891 triples (for 196 questions), an average of 4.5 triples per question.

3.4 Question Answering Routines

For TREC-8, a database of documents was created for each question, as provided by the NIST generic search engine. A single database was created for the questions themselves. The question-answering consisted of matching the database records for an individual question against the database of documents for that question.

The question-answering phase consists of three main steps: (1) coarse filtering of the records in the database to select potential sentences, (2) more refined filtering of the sentences according to the type of question, and (3) scoring the remaining sentences based on matches between the question and sentence database records. The sentence were then ordered by decreasing score for creation of the answer file submitted to NIST.

3.4.1 Coarse Filtering of Sentences

The first step in the question-answering phase was the development of an initial set of sentences. The discourse entities in the question records were used to filter the records in the document database. Since a discourse entity in a record could be a multiword unit (MWU), the initial filtering used all the individual words in the MWU. The question and sentence

discourse entities were generally reduced to their root form, so that issues of tense and number were eliminated. In addition, all words were reduced to lowercase, so that issues of case did not come into play during this filtering step. Finally, it was not necessary for the discourse entity in the sentence database to have a whole word matching a string from the question database. Thus, in this step, all records were selected from the document database having a discourse entity that contained a substring that was a word in the question discourse entities.

The join between the question and document databases produced an initial set of unique (document number, sentence number) pairs that were passed to the next step.

3.4.2 Refinement of Viable Sentences

The second step of the question-answering process applied more detailed screening of the sentences. This screening involved the application of criteria based on the type of question.

As indicated above, one record associated with each question contained an unbound variable as a discourse entity. The type of variable was identified when the question was parsed and this variable was used to determine which type of processing was to be performed during the sentence refinement step.

The prototype system recognized six question types (usually with typical question elements): (1) **time** questions ("when"), (2) **location** questions ("where"), (3) **who** questions ("who" or "whose"), (4) **what** questions ("what" or "which," used alone or as question determiners), (5) **size** questions ("how" followed by an adjective), and (6) **number** questions ("how many"). Question phraseology not envisioned during the prototype development (principally questions beginning with "why" or non-questions beginning with "name the ...") were assigned to the **what** category, so that question elements would be present for each question.

Some adjustments to the question type were made just prior to the refined filtering. Specifically, it was recognized that questions like "what was the year" or "what was the date" and "what was the number" were not **what** questions, but rather **time** or **number** questions. Other phraseological variations of questions are likely and could be made at this stage.

In general, the functionality for the screening step involved elimination of sentences from further processing (based on criteria described below) and initialization of the data structure for holding a 50-byte answer. An initial score (of 1000) was assigned for each sentence during this process. And, the number of viable sentences was limited.

1. Time Questions - The first criterion applied to a sentence was whether it contained a record that has a **TIME** semantic relation. The parser has specific mechanisms for recognizing prepositional phrases of time or other temporal expressions (e.g., "last Thursday"). During the analysis of the parser output, the database records created for these expressions were given a **TIME** semantic relation. After screening the database for such records, the discourse entity of such a record was then examined further. If the discourse entity contained an integer or any of its words were marked in the parser's dictionary as representing a time period, measurement time, month, or weekday, the discourse entity was selected as a potential answer.

2. Where Questions - Each sentence was examined for the presence of "in" as a semantic relation. The discourse entity for that record was selected as a potential answer.

3. Who Questions - There was no elimination of sentences for these questions. All sentences were continued to the next step. A potential answer was developed by searching for a record that had the same governing word as that of the unbound variable. (For example, "who created ..." would show "create" as the governing word; a match would be sought for a sentence record with "create" as the governing word.) The head noun of the discourse entity would be the potential answer.

4. What Questions - There was no elimination of sentences for these questions. All sentences were continued to the next step. A potential answer was developed by searching for a record that had the same governing word as that of the unbound variable. The discourse entity would be the potential answer.

5. Size Questions - The first criterion applied to a sentence was whether it contained a record that has a **NUM** semantic relation. The parser has specific mechanisms for recognizing numbers. During the analysis of the parser output, the database records created for these expressions were given a **NUM** semantic relation. If these expressions were followed by a noun, the noun would be captured as the

governing word. After screening the database for NUM records, the governing word of such a record was then examined further. If any of the words of the discourse entity were marked in the parser's dictionary as representing a measure, a unit, or a measurement size, the discourse entity, a space, and the governing word were constructed as a potential answer.

6. Number Questions - The same criterion as used in size questions was applied to a sentence to see whether it contained a record that has a NUM semantic relation. In these cases, the number itself (the discourse entity) was selected as the potential answer.

3.4.3 Sentence Scoring

Each sentence that passed the screening process of the previous step was assigned a base score of 1000 and was then evaluated for further correspondences to the question database records. Each record of the question database was examined in relation to each record for the sentence in the document database. Points were added or deducted based on correspondences.

If the discourse entity in the sentence record is a proper or complete substring of the discourse entity in the question record, 5 points are added when the semantic relation or governing word match completely. Five points are deducted if the match is not complete.

If the question discourse entity is an MWU, each word of the MWU is examined against the discourse entity in the sentence record. If a word in the MWU is a substring of the sentence discourse entity, 5 points are added to the score. If the last word of the MWU is a substring of the sentence discourse entity (generally corresponding to the head noun of the MWU), 20 points are added. When we have a substring match, we also test the semantic relation and the governing word of the two records, adding 5 points for each match.

In general, then, points are added because of matches in the semantic relation and governing word fields, but only when there is at least a partial match between the discourse entities of the two records. Thus, the focus of the matching is on the structural similarity between the question records and the sentence records, i.e., on whether the discourse entities participate in the same type of semantic relations with the same governing word. Many of the sentences

passed to this step will have minimal changes to their scores, while those that match on structural similarity will tend to separate out relative to other sentences in the documents.

After scores have been computed for all sentences submitted to this step, the sentences are sorted on decreasing score. Finally, the output is constructed in the desired format (for both 50-byte and 250-byte answers), with the original sentences retrieved from the documents.

4. TREC-8 Q&A Results

The official score for the CL Research 250-byte sentence submission was 0.281. This means that, over all questions, the CL Research prototype provided a sentence with the correct answer as its 4th choice. This compares to an average score of 0.332 among the 25 submissions for the TREC-8 Q&A track (i.e., a correct answer in the 3rd position). In examining all the answers submitted by the various teams, the CL Research prototype was one of only two teams that submitted full sentences, as opposed to a 250-byte window around an answer.

The CL Research prototype submitted sentences containing correct answers for 83 of the 198 questions. Compared to the median scores for the 198 questions, the CL Research prototype was better than the median for 40 questions, equal for 109 questions, and less for 49 questions. Since CL Research did not provide a correct sentence for 115 questions, this means that for 66 questions, the median score among the 25 participating systems was unable to provide a correct answer. Finally, the CL Research prototype equaled the best score for 46 questions and the worst score for 115 questions (i.e., the questions where CL Research did not provide a correct answer).

The CL Research prototype performed better than the average score of 0.332 for 56 questions. On these questions, the average score was 0.447 (that is, a correct answer was given as the 2nd ranked answer over all participating systems). Thus, the questions for which the CL Research prototype provided correct answers were in general easier than the remaining questions. However, among these questions, 39 were easier than the average and 17 were more difficult than the average of 0.332. In other words, the CL Research prototype did not just answer the easier questions, but was able to answer some of the more difficult questions as well.

5. Analysis

The results achieved by the CL Research prototype seem to indicate that the general approach of matching relational structures between the questions and the documents is viable. The prototype selected 937 sentences and at least 83 correct sentences out of over 63,000 sentences in the source documents, so the approach clearly performed much better than random. Since the prototype was an initial implementation, focused primarily on just providing a set of answers, without any evaluation of alternative approaches and without inclusion of several generally available research findings (e.g., named-entity recognition, time phrase computations, and coreference resolution), the approach seems to have much promise.

Even with the claimed level of performance, however, it seems that the official results significantly understate the viability of the general approach in the prototype. This statement is based primarily on the fact that only the top 10 documents were used in an attempt to answer the questions, when frequently an answer did not appear in any of these documents. There are several other simple changes (such as resolution of relative time phrases to a specific date, where the appropriate phrase was identified in the prototype as one of the submitted answers) which would result in a higher score.

Overall, based on post-hoc analysis of the cases where the CL Research prototype did not provide the correct answer, it is estimated that a more accurate overall score is approximately 0.482. (This estimate is based on post-hoc analysis of 25 percent of the questions where no correct answer was provided.) The reasons justifying this estimate are detailed below.

1. Cutting off sentences - For three questions, the limitation to 250-byte strings cut off the portion that would have recognized by NIST evaluators as correct. In each case, the appropriate sentence was ranked first, adding 0.015 to the overall score.

2. Inclusion of document containing answer - Post-hoc analysis revealed that for one-third of the questions, the answer was not in the top 10 documents included in the database for the question. When a document containing the answer was added to the database, correct answers were identified in two-thirds of the cases, with an average inverse rank of 0.320, adding 0.153 to the overall score.

3. Relative time resolution - One-fifth of the questions answered incorrectly required resolution of relative time phrases ("last Thursday," "today," "two years ago"). The functionality for this time resolution is essentially already present in the CL Research prototype and the document date necessary for this computation is contained in the document databases. The average inverse rank for the sentences provided in the prototype results is 0.292, adding 0.033 to the overall score.

There are some considerations in addition to the above that also would portray the CL Research prototype more favorably, but for which no immediate estimate of improvement in the overall score is claimed. For 6 percent of the incorrect answers, a sentence containing the correct answer was generated and was tied with an answer that was submitted. However, because the sentence was not generated in a timely order, it was not submitted.

The correct answer was also generated for another 20 percent of the cases where no correct answer was provided, but the appropriate sentences were ranked lower than those submitted. In another 17 percent of the cases with no correct answers, the answer required coreference resolution from one of the sentences submitted. About 10 percent of the cases involved incomplete creation of document database entries due to bugs in the parser or in the mechanisms for extracting database records; it is not yet clear what effect correcting these difficulties would have. (These difficulties resulted in no sentences being submitted for 6 questions.)

Further examination of the results is necessary to understand the factors contributing to success and failure. A first analysis investigated the relation of the question parsing and database construction to whether a question was answered correctly. About 65 percent of the correct answers had no problems with parsing and database construction for the question, compared to 51 percent of questions with incorrect answers. Only 20 percent of the questions with correct answers had parsing problems, compared to 25 percent for those with incorrect answers. About 8 percent of the questions with correct answers involved questions not overtly recognized by the prototype (where a default **what** question type was used), compared to 11 percent of the questions with incorrect answers. Finally, 7.5 of the questions with correct answers had words unknown to the parsing dictionary, compared to 4.5 percent for questions with incorrect answers.

There were no obvious correlates for the scores. For questions with correct answers, the scores were tabulated by the number of database records generated by the questions (which ranged from 2 to 10, with an average of 4.4 records). Higher scores would have been expected for those questions with a higher number of records, but instead the average scores were about the same across the range. Similarly, the average scores for the top 5 rank answers were nearly identical for questions answered correctly and questions answered incorrectly. The average scores for the submitted answers were weakly correlated with the difficulty of the questions (0.18) reported by NIST. However, the correlation was lower (0.11) for questions answered correctly and much higher (0.25) for the questions answered incorrectly. This result is somewhat paradoxical.

Further detailed analysis is necessary to get at the most significant contributors to the scores. Heuristics were used in developing the scoring mechanisms. At this time, these heuristics have not been evaluated, either for identifying the valid or invalid contributions to the scores or for evaluating the weighting scheme.

6. Anticipated Improvements

The immediate possibilities for improvements are many and the possibilities for exploration are quite diverse. In addition, there are opportunities to be explored for integrating the prototype within more generalized search engines. Finally, the prototype can be examined for suitability for use with specialized textual databases.

The immediate improvements are evident from the analysis indicated above: (1) dealing with problem cases where answers weren't generated because of problems with parsing the questions or extracting appropriate database triples from the questions; (2) addition of time phrase analysis routines; (3) extension of the question types handled by the system; and (4) problems in extracting the document database triples arising from the parser or extraction routines.

Less immediate, but straightforward extensions can be gained by incorporating (1) coreference resolution techniques and (2) named entity techniques. The database extraction routines constituted a minimal implementation. These can easily be extended by further analysis of the parse output.

At the next level of complexity, the database extraction techniques require further refinement and extension. The semantic relations used in the prototype can be enhanced, particularly beyond the use of specific prepositions as the characterizing element. The reliance on specific prepositions is likely to have reduced matches; generalizing specific prepositions to more general semantic relations would yield better matches that would not be dependent on specific phraseology. The next step along these lines would involve incorporation of better discourse analysis techniques (such as text summarization research (Mani & Bloedorn, 1999)) for tying together records in the document databases.

Along the same lines, the prototype could be improved by incorporating techniques from lexical cohesion and lexical chain research to tie database records together (Green, 1997). This would specifically involve use of semantic network data (such as is present in WordNet or MindNet), particularly to link synonymic and hypernymic phraseology. Larger dictionaries would also be of some use.

Finally, the mechanisms in the prototype can be improved. Further post-hoc analysis will likely lead to better analysis and selection of sentences. The mechanisms for examining the selected sentences (during the analysis of specific question types) were also somewhat minimal in the prototype; further analysis is likely to yield improvements. Evaluation of the scoring mechanisms (understanding why appropriate sentences received lower scores than higher ranked sentences and understanding the contribution of the individual mechanisms) will also likely lead to improvements.

Since the prototype did not include a general search engine, the best interface with such systems is unknown. In addition, there are many applications that attempt to answer questions from specialized databases (such as FAQ databases, automatic message responders, and help files). There are also many specialized textual databases (historical records or genealogical databases). It seems that the prototype can work immediately with more or less static text databases, but in all these instances should also be able to take advantage of search functionality already included in such systems.

Some caveats are necessary in considering the results of the CL Research prototype and the possible improvements. Many of the questions in the TREC-8 Q&A track can possibly be better answered by simple

lookup in dictionaries (including those that contain small amounts of encyclopedic information). Also, it appeared as if the phraseology of the questions frequently was very close to the answers in the text. The extent to which these considerations affect results needs to be determined.

7. Summary

The CL Research prototype system was reasonably successful in answering questions by selecting sentences from the documents in which the answers occur. The system generally indicates the viability of using relational triples (i.e., structural information in a sentence, consisting of discourse entities, semantic relations, and the governing words to which the entities are bound in the sentence) for question-answering. Post-hoc analysis of the results suggests several further improvements and the potential for investigating other avenues that make use of semantic networks and computational lexicology.

References

- Fellbaum, C. (1998). *WordNet: An electronic lexical database*. Cambridge, Massachusetts: MIT Press.
- Green, S. J. (1997). Automatically generating hypertext by computing semantic similarity [Diss], Toronto, Canada: University of Toronto.
- Litkowski, K. C., & Harris, M. D. (1997). *Category development using complete semantic networks*. Technical Report, vol. 97-01. Gaithersburg, MD: CL Research.
- Mani, I., & Bloedorn, E. (1999). Summarizing similarities and differences among related documents. *Information Retrieval*, 1(1), 35-68.
- Richardson, S. D. (1997). Determining similarity and inferring relations in a lexical knowledge base [Diss], New York, NY: The City University of New York.

A Connectivity Analysis Approach to Increasing Precision in Retrieval from Hyperlinked Documents

Cathal Gurrin & Alan F. Smeaton

School of Computer Applications
Dublin City University
Ireland
cgurrin@compapp.dcu.ie

Abstract

Within the last few years very little has been made of the usefulness of Connectivity Analysis to Information Retrieval on the WWW. This document discusses hyperlinks on the WWW and our experiments on the exploitation of the immediate neighbourhood of a web page in an effort to improve search results. In order to test the hypothesis that Connectivity Analysis can increase precision in the top ranked documents from a set of relevant documents, we developed a software application to generate and re-rank a query relevant subset of the WT2g Dataset. We discuss our software in depth and the problems that we encountered during development. Our experiments are based on implementing a number of re-ranking formulae, each of which tests the usefulness of different approaches to re-ranking a set of relevant pages, ranging from basic context analysis (inLink ranking) to combined content and context analysis approaches.

1. Introduction

Within the last few years very little has been made of the usefulness of Connectivity Analysis to Information Retrieval on the WWW. Connectivity Analysis as an IR technique is based upon the identification and exploitation of latent linkage information inherent in the structure of the WWW. In fact, as the WWW grows it becomes increasingly difficult for standard IR approaches to operate effectively. However hypertext links, the building blocks of Connectivity Analysis are constantly increasing in number and becoming more important.

Our approach outlined in this paper is to utilise connectivity information to improve the ranking of results from web search. In effect we are attempting to improve the precision of the top documents returned from a search as it has been shown that up to 85% of users only look at the first page of search results. We do this by developing a number of approaches each of which is based around re-ranking a set of 'relevant' documents by one of a number of popularity ranking formula. In so doing we distinguish between different types of links to be found on the WWW of today, even though only one syntactic type of link is supported by HTML.

2. Background and Hypothesis to Test

An author writing a WWW document will create different types of hyperlinks (or edges) between documents, even though HTML supports only one syntactic type of hyperlink (represented in HTML by the <a> tag). In fact web page authors will most probably not be aware of the significance of the different link types that they are creating. In [4] Spertus discusses hyperlinks and varieties of hyperlink information, based on information mined from identifying the target of each link. She states that much human thought has gone into creating each hyperlink and labelling it with anchor text, which forms the basis of the approaches taken in [1,2,3].

2.1 Analysis of Link Structure

The WWW in its present form is said to consist of approximately 1 Billion web pages and up to 10 Billion associated hyperlinks. This ratio of approximately 1:10 can easily be viewed in the context of the 2GB WT2g DataSet we use in the TREC-8 web track. Here we have 247,491 individual HTML documents with a total of 2,254,515 links (ratio is 1 : 9.1) from these documents to other documents. Henzinger & Bharat in [5] discuss a Connectivity Server, developed in the AltaVista Labs, that stored 1 Billion edges related to 100 million nodes, generated from AltaVista crawls. This Connectivity Server would have represented the adjacency matrix $A(G)$ for the AltaVista WWW index. In the course of our research we had to develop a Connectivity Server to handle queries for our own search software. Details will follow below.

Generally on the WWW we can separate links into one of two types based on their intended function when being created:

- **Structural** (upward, downward or crosswise) links that link separate nodes within a particular domain. They exist to aid the user in navigating within a domain, or web site.
- **Functional** (content, or outward) links on the other hand link nodes in different domains (across web site boundaries). They can be seen to mostly link from a source node to a target node that contains similar and, in the author's opinion, useful information. The author of a particular node is likely to have found the target node useful and related to the concept explored in the source node, and for that reason created the link.

In the course of this research we are more interested in the latter type of link as opposed to the former. We view all nodes within a domain as having being written/developed by the one author and representing the ideas of a single individual or organisation. Consequently the former can not be seen as a source of useful information for Connectivity Analysis because we can not allow individuals to directly influence the popularity of their own web pages (spamdexing). See [4] for a more detailed discussion of Structural links and their uses. Utilising Functional links can give us a means to judge the popularity of a particular page to WWW page authors and consequently we can generate a popularity rank for the page. When extracting information from hyperlinks on the WWW, we assume two properties inherent in hyperlinks from [6], these are:

- A link between two documents implies that the documents contain related content.
- If the documents were authored by different people, then the first author found the second document valuable.

Additionally, one can assume that a page with a large number of Functional *outLinks* will act as some type of index page and be a source of links to content that the author of the index (source) page found relevant and useful. Similarly a page with a large number of Functional *inLinks* could be seen as a popular page precisely because a lot of people have found it useful, and consequently linked to it. One could take this a step further and conclude that a document on a topic such as *Formula 1 Motor Racing* which has *inLinks* from a number of other documents relating to a similar topic, would be more useful to a user looking for information on *Formula 1* than a document which may have a number of *inLinks* from more diverse sources. It is this belief that has influenced our TREC approach and led Kleinberg [7], Brin & Page [8], and others to their conclusions as to the usefulness of Connectivity Analysis as an alternative/complementary approach to general IR on the WWW.

Kleinberg [7] describes the WWW as consisting of multiple communities of documents based around themes. He puts forward the concept of *Hubs* and *Authorities*. A *Hub* page is a page that acts as a source of links to related content. A *Hub* page will typically contain many more *outLinks* than *inLinks* and may be seen as a form of index page. An *Authority* page on the other hand, is regarded as a source of information (an authority) on a subject and is pointed at by *Hub* pages. *Hubs* and *Authorities* exhibit a mutually reinforcing relationship, that is, the higher the quality of the nodes that point to a page, the higher the quality of that page, and consequently this in turn increases the quality of any pages that link into that page. His experiments show that broad queries on broad topics can benefit greatly from Connectivity Analysis where his software (HITS) can select the most popular (best-connected) document from within a linkage-based community on the WWW. In addition, we can categorise results from a query into linkage based clusters by implementing a form of eigenvalue decomposition on a connectivity matrix generated from the immediate neighbourhood of the Result Set which has been generated in response to a query.

2.2 Representation of the Link Structure of the WWW

A natural method of representing the WWW is as a directed graph with (a finite, non-empty set of) nodes representing documents and edges representing the links between these documents. For a detailed explanation of this concept see [9] where Adamic describes the WWW in terms of a small world graph. An edge represents a directed (source to target) link between two nodes because the HTML hyperlink is uni-directional. Within this graph we can select a sub-graph (G), consisting of a subset of the graph's nodes and a subset of the graph's edges. Before we can work with a graph such as G we need a method of representing it. The adjacency matrix of the graph is a suitable method of doing this. In the adjacency matrix $A(G)$ for G , the entry in row i and column j is 1 if the nodes i and j are joined by an edge and 0 otherwise. Of course an adjacency matrix for the entire WWW at a given point in time would be enormous, sparse and very difficult to model. Our representation of connectivity information as an adjacency matrix opens up whole new mathematical approaches to IR on the WWW, see [7]. Assuming the adjacency matrix $A(G)$ can represent the *outLinks* of the set of documents, the vast majority of *inLinks* into these documents can be represented by $A^T(G)$ which is the transpose of the matrix $A(G)$. A Connectivity Server, similar to the one described in [5], would operate using an up-to-date adjacency matrix for all the documents indexed by it.

A further implementation of graph theory would allow for the generation of a weighted graph representing the WWW (or a subset thereof) in which each edge would have an associated weight. A number of formulae that we have implemented in our research require the use of weighted edges in a sub-graph of relevant documents. This sub-graph having been generated by implementing a content analysis search on the dataset to select query relevant documents. By using a weighted graph we can regulate the proportional relevance of link types as we find links to be of varying importance, depending on the context in which they were created.

The hypothesis we set out to test was whether, given a set of 'relevant' documents, re-ranking them by a popularity measure would improve the precision of the top results, and consequently populate the higher rank positions in a result set with a larger number of higher quality/popular documents. Our approach is based on the assumption that, by implementing a conventional text-based search on the dataset, a small subset of nodes that are relevant to the topic in question will be generated. The execution of various formulae on this small subset will then result in applying higher weights to the most popular nodes and therefore increase the ranking of the related documents in the result set.

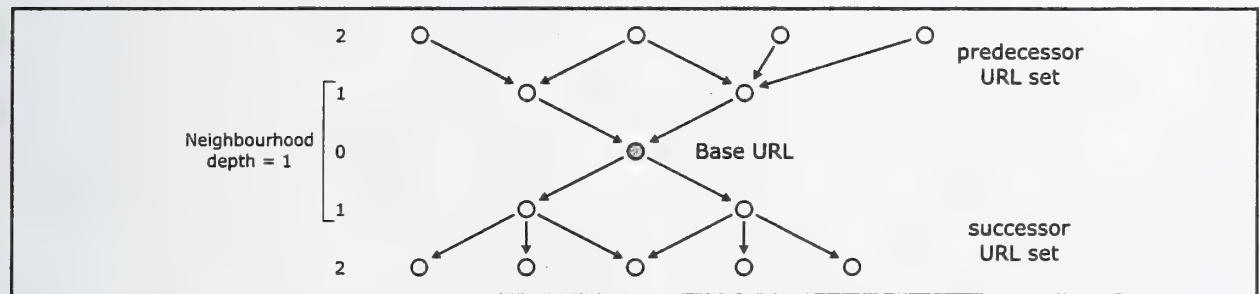


Figure 1: Link Types

Our approach was developed in order to provide higher quality results to a search by focusing on the immediate neighbourhood of the document in the WWW graph, to a depth of 1, see Figure 1. In [1] Li describes the Hyperlink Vector Voting method which ranks a document on the basis of the number of hyperlinks pointing into it (in its immediate neighbourhood) and uses the hyperlinks anchor text as an indication of the semantic content of the target document. He compares the approach with 'voting' for results on the WWW, i.e. a document's popularity & content is dependent on other authors, and not directly on the author him/herself. For additional approaches to suggesting the content of a document, by looking at the anchor text from *inLinks* into the document, see [2,3].

3. System Overview

As previously stated, the reason for having a Web Track in TREC-8 was to investigate whether Connectivity Analysis is a useful aid to web search. In order to test whether the associated hypothesis is true, we developed a software application which would produce results based on conventional Content Analysis (the baseline result) and then re-rank those results based on a number of related Connectivity Analysis approaches. Our approach was based on using the WT2g dataset, consisting of 247,491 HTML documents at 2GB storage requirements. Although we felt that the 100GB collection would be more useful as a research tool, we didn't have the storage capacity available to handle such a large dataset at that time.

3.1 DataSet Preparation

The TREC dataset and the connectivity data required pre-processing before they were of use to our software. The 247,491 HTML pages were contained in 1081 separate text files. Each HTML page had to be extracted and stored on disk before our Content Analysis software would index it. Consequently we developed an application to extract all 247,491 HTML pages from the source. These pages were given a name based on the TREC document ID (e.g. "WT04-B21-15.html") and saved to disk. As these files were being generated, we extracted a small amount of information from each document, which we stored in a database. The information we extracted consisted of:

- Document ID (internal unique representation for each document)
- Original document URL
- Document title (where available)

This information would later be used to generate intuitive results for each query, see Figure 4 for an example. This information was stored in a SQL Server Database and accessed at query execution time. It was found after testing that the Database Server we used, although it was on an under-powered PC, was capable of handling over 100 queries per second, which proved sufficient for our needs.

Once all pre-processing of the source files had been completed, files written to disk and our Database indexes generated, our Content Analysis software indexed the files. To avoid having to develop our own Content Analysis software, we used AltaVista Discovery. Discovery is a desktop term-based search application provided by AltaVista and freely downloadable from their web site. Discovery usually indexes Word, Excel, Email files etc... but also is capable of indexing HTML files stored locally on disk. Consequently it suited our needs for a term-based search engine that was capable of accepting queries and returning sets of highly-scored documents for additional processing. It took Discovery in the region of 160 hours to complete the processing of the dataset. Much of this processing was done in stages at night and at weekends as the PC was being used during the day to develop the search software. Once completed the index files generated by Discovery amounted to 1GB in size, approximately half the size of the dataset itself. We found a number of limitations with using Discovery for this task:

- Discovery will only accept requests from the local PC on IP address 127.0.0.1. Consequently all searches were run on the computer that contains Discovery, the Discovery index and the Dataset. Since our official runs, we have developed a Server to handle queries from a computer other than the one Discovery is installed on and pass back the result. Discovery is currently installed on a number of computers, each of which indexes a different version of the dataset.
- Discovery would only list the top 200 documents in response to a query. We found no way around this limitation and consequently our results only ever contained a maximum of 200 documents, even when more were deemed relevant. We avoided expanding the set of 'relevant' documents using the neighbourhood of these documents as in [7] due to the fact that our Result-Set already contained too many irrelevant documents and that by expanding this set would incorporate even more irrelevant documents and led to severe 'topic-drift' problems.
- Discovery was slow at providing responses to queries. Most queries took in the order of 15 seconds to produce content-only results for a query. Using recently acquired hardware we have reduced this time to below 2 seconds per query.

Due to latency problems encountered when using the on-line Connectivity Server we decided to develop a local Connectivity Server using the downloadable connectivity data provided by the Web Track organisers. Once it was made available we developed a simple application to process the connectivity files and insert the linkage data

into another SQL Server Database which acted as a local Connectivity Server. From our point of view it provided us with a version of the A(G) Adjacency Matrix for the entire WT2g Dataset, both for *inLinks* and *outLinks*. The weight of each edge between nodes was set to a default value of 1. The connectivity server worked by accepting a document id *i* and returning a list of all document id's that pointed to *i* or are pointed at by *i*. The internal structure of the Connectivity Server is very simple, consisting of simply source node id and target node id pairs to represent each link between two documents.

The software was implemented in JAVA under Windows NT, and the following two computers were available for our use at that time:

- PII 350, 64MB RAM, 8GB HDD, Windows NT 4

This computer provided all the processing power for the search session, accepted queries, ran Discovery, processed the result set and generated results for the user.

- P 120, 32MB Ram, 1GB HDD, Windows NT Server 4 & SQL Server 6.5

This computer provided permanent storage for document indexes and acted as a Connectivity Server.

These machines have since been replaced by a suite of networked PCs, which will be dedicated to our further work in this area.

3.2 Query Generation Technique

We took two approaches to developing queries from the topics 401-450. The first approach (submitted to TREC as official runs) was based on the title of the topic alone. However we felt this would adversely influence the quality of results produced in that a more advanced automatic, or manual technique would produce higher quality queries and consequently Result-Sets containing a higher proportion of relevant documents, as well as being more representative of a normal user's query on the WWW. Accordingly we developed a set of queries which were manually generated from the Topics and ran tests using these queries, the results of which are in this document. We found that the manually generated queries produced results that increased the precision of the baseline (content-only results) for the top 30 documents, which is what we are interested in when researching web search techniques. Figure 2 provides exact values for content-only precision at 5 to 1000 documents for both modified and unmodified queries. Use of the modified queries, as opposed to unmodified queries increases the percentage of relevant documents in the 'root-set' from 8.81% relevant to 22.29% relevant, but decreases the overall number of documents returned from 7964 to 1857, thus giving smaller, but more focused content-only result sets.

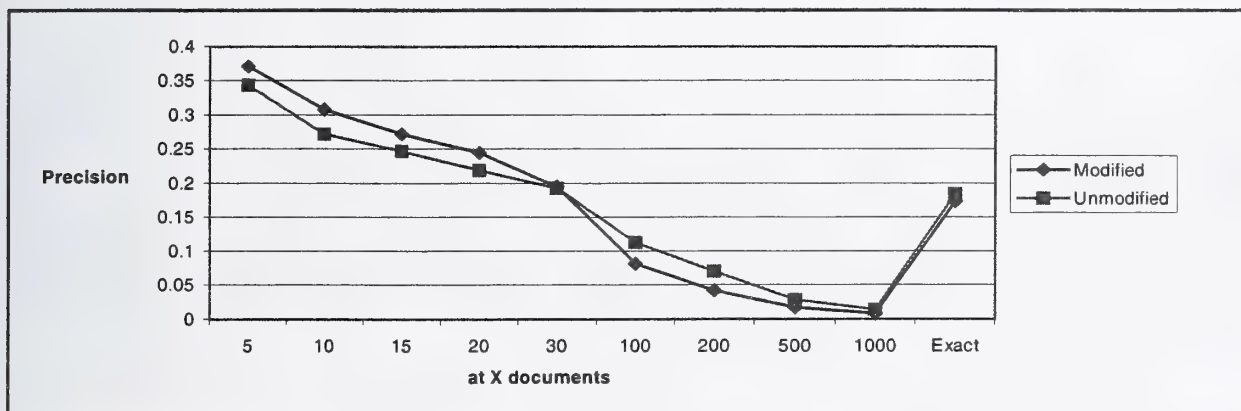


Figure 2: Baseline relevance depending on Query

4. Experiments

4.1 Re-ranking

In order to test the usefulness of various Connectivity approaches to web search our experimental software generates a content-only result set for a query. This is done by sending the query to our Discovery server, reading back and parsing the results. This baseline set of documents, the 'Result-Set', can be expected to consist of documents

relevant to the query, or at least highly scored documents, although this was not as successful as we had hoped. We developed simple re-ranking schemes to re-rank the this set of relevant documents which would allow for easy visualisation of the differences between a content only approach and a context analysis approach to Web Search. Consequently the process of executing a search on our software consists of 2 stages, a content analysis stage and a context analysis stage.

4.2 Stage 1 - Content Analysis

Each search firstly consists of a content analysis stage performed by Discovery on the test collection. The user enters a query, which is passed to the Discovery Server (an application running on a computer that is running Discovery), which in turn, sends the query (unmodified) as an HTTP request to Discovery (on the local machine), and passes the generated result back to the client computer. Discovery queries it's indexes and returns the top 200 results (max) to the query. The result returned from Discovery is then parsed to produce a set of up-to 200 documents (the Result-Set), or baseline result. The benefit of this is that a set of documents is generated which is assumed to be relevant to the query. Overall we found that, at best, only 22.29% of the Result-Set returned from Discovery was deemed relevant to the queries in topics 401-450. This Result-Set is then available for further processing. This completes Stage 1 and the results are written to disk. In this way we provide content-only results to the queries, which were submitted to TREC as an official content-only run (*dcu99c01*).

4.3 Stage 2 - Context Analysis

Stage 2 consists of a context analysis phase that re-ranks the Result-Set generated in Stage 1, based on a number of popularity ranking formulae. Each formula was developed as a separate algorithm (each of which we refer to as a numbered variation), which could be executed on the Result-Set to generate results. In all we developed 7 variations of the re-ranking algorithm (and associated formulae), each of which could be executed independently, i.e. without affecting the execution of, or results produced by any other variation. In all of the variations, if the P_n (popularity of document_n) scores of a number of documents are equal, the original document order given by Discovery is used to decide on the final ranking given to these documents.

In Connectivity Analysis we generally assume that the more popular a document is, the more *inLinks* that document will have on the WWW, hence:

Variation 1

$$P_n = \sum \text{inLinks}_n$$

In this case the P_n score is based purely on the number of *inLinks* to document_n. This we developed as the first variation that produced results for further evaluation.

However this approach is rather too simplistic. What would happen if a site such as *Yahoo* or *Microsoft* were to be contained in the Result-Set. Regardless of the topic in question these sites would be ranked highest due to the number of *inLinks* associated with them. Although none of these sites are represented in the WT2g dataset, we had to take this possibility into account. Consequently we would recommend limiting and in our research did, limit the number of *inLinks* to a maximum of 50, resulting in:

Variation 2

$$P_{n(0..50)} = \sum \text{inLinks}_n$$

In addition to ranking by *inLinks*, we wanted to recognise the potential of a document to act as a source of links to further information (a *hub* document). Accordingly we developed a third variation that took the number of *outLinks* into account as well as the number of *inLinks*. Once again we limited the number of *outLinks* considered for any one particular document, however this time we set the limit to 20. This was in order to avoid some large index type documents swamping the results. In addition we felt that *inLinks* would be far more relevant in generating a popularity score for documents than *outLinks* so we set the number of *inLinks* to be weighted 4 times the weight of the *outLinks*, giving the following formula:

Variation 3

$$P_n = (4 * \sum \text{inLinks}_n (0..50)) + (1 * \sum \text{outLinks}_n (0..20))$$

Increasing the weighting of *inLinks* would mean that our sub-graph would have constituent edges weighted at 4 if their targets were the node in question, but any edges with their source being the node in question would retain the default weighting of 1. This formula that we called *Rerank* was entered as one of our linkage-based runs to TREC (*dcu99l01*). This approach (assuming a Result-Set set of 200 documents) required 400 SQL Queries (this could have been reduced to 200 queries, if necessary) and consequently required 4 seconds or so to complete. The processing time required for *Rerank*, aside from the SQL queries is negligible, just the calculation of P_n 200 times.

Building on the third variation, we developed the fourth that combined both context and content analysis in Stage 2. Instead of simply re-ranking each document based on a popularity formula we allowed the ranking given in Stage 1 by AltaVista Discovery to influence P_n up to the same extent as *inLinks*. We felt that this should help the ranking process to keep the most syntactically relevant documents (as chosen by Discovery using its term matching) near to the top of the results. The formula we used can be seen below:

Variation 4

$$P_n = (2 * \sum \text{inLinks}_n (0..50)) + (1 * \sum \text{outLinks}_n (0..20)) + (\text{discWt}_{(200..0)} / 2)$$

This formula was called *RerankAdv* and required one additional modification before it was finalised. The type of links used in calculation of P_n for each document was also taken into account.

As we discussed earlier, it is intuitive that we only want to count Functional links and ignore Structural links altogether from the re-ranking equation. Consequently we had to check each *inLink* and *outLink* to determine its type. This was done using the domain strings of the node in question. We selected the lowest level substring of the domain (e.g. www.microsoft.com) for the base URL and the target URL and compared them. If the two were the same URL then we recorded the match and ignored the link as it was considered to be Structural. We felt that this approach was more valuable than ignoring all links to ***.microsoft.com as it was felt that a URL such as research.microsoft.com was not to be considered authored by the same author as www.microsoft.com. The one author per domain assumption would not apply in this case. Had we taken this strict Functional-only approach we would have been left with a subgraph of the Connectivity Data, but with only Functional (across domain) links remaining.

Desirable as the approach to only include Functional links may seem, we found that the 2GB dataset prohibited us from fully implementing this approach. This was due to a lack of Functional links between documents in the dataset. To overcome this problem and to produce acceptable results we had to allow Structural links to exert some influence on the P_n score of each document. In a real implementation of this approach on the WWW this would not be the case as re-ranking by Structural links serves to increase the ranking of documents from large, well interconnected (within their domain), web sites. Consequently we weighted Functional *inLinks* at 4 times the weight of Structural *inLinks* and limited the total link score (inLink_n) for each document to a maximum of 50. This replaced our subgraph with a weighted subgraph with a select few edges (Functional) having a weight of 4, with the rest having the default weight of 1. The following formula was used to calculate the *inLink* score of a node:

$$\text{inLink}_{n(0..50)} = ((4 * \sum \text{funct}_n) + \sum \text{struct}_n)$$

This fourth variation, *RerankAdv*, was submitted as one of our preliminary runs to TREC (*dcu99l02*). In addition to these variations another two were developed in order to clarify the benefits (if any) of re-ranking by both Functional and Structural *inlinks*. The fifth was to re-rank the Result-Set based on the number of Functional *inLinks* alone and the sixth was to re-rank based on the number of Structural *inLinks* alone:

Variation 5

$$P_n = (\sum \text{funct}_n)$$

Variation 6

$$P_n = (\sum \text{struct}_n)$$

The seventh and final variation we developed was an attempt to overcome the lack of Functional links in the dataset. Our approach was to utilise the connectivity information for each document that would be contained in the WWW as a whole. The problems of using live WWW connectivity data on the 2GB Dataset were threefold:

- many of the documents were no longer in existence as the documents from which the Dataset had been generated had been spidered in early 1997.
- many may not have been indexed by the Connectivity Source we were querying.
- a number would have had their content modified since spidering and thus the current WWW connectivity information would be considered invalid.

In order to get around this problem we replaced the actual URL of the document with the URL of its domain. This shortened URL was then used to query the AltaVista search engine using the "link:URL" query that returns the number of and actual *inLinks* into the document in question. We then ranked by the popularity of the main page (index.html) within each domain.

Variation 7

$$P_n = \sum \text{inLinks}_m \text{ (m = lowest level of domain string)}$$

5. Results

We entered three of our experimental approaches as preliminary runs into TREC in August 1999, two of which were linkage-based and the third a content-only run based on the results of Discovery. The linkage-based runs (*Rerank* – variation 3 and *RerankAdv* - variation 4) were detailed earlier in this document. Two of our runs were added to the pool for relevance judgement purposes (*dcu99c01* & *dcu99l02*). Additional experiments that we ran after submitting our official runs found little or no improvement in precision when incorporating Connectivity Analysis in the ranking process in the majority of cases. Our experiments find that using Functional links from within the supplied connectivity data provides the best results of all the Connectivity Analysis approaches outlined in this document. However the number of Functional links within the dataset are extremely limiting at less than 1% of the total links and therefore only a limited number of documents would be re-ranked from any one query. Consequently we are currently unable to come to any definite conclusions as to the benefit of re-ranking by Functional *inlinks*. We hope that the 10GB dataset next year will contain a higher percentage of Functional links. Figure 3 below shows the precision at 5 - 1000 documents returned from running the modified queries on WT2g.

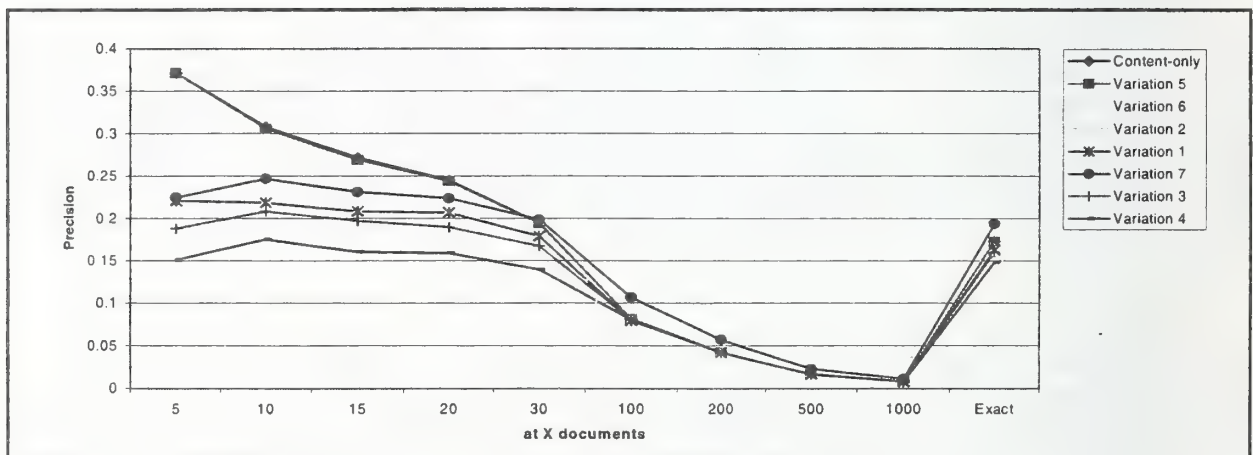


Figure 3: Precision at x Documents

We found that re-ranking by Structural *inLinks* is not viable either as this would only serve to rank highest documents from within large, well inter-connected domains. Due to the lack of Functional links in WT2g, we found that the scores for Variation 1, Variation 2 and Variation 6 are almost identical. Variation 1 and Variation 2 do not distinguish between Functional and Structural link types, yet the P_n score associated with them is almost identical to the P_n score of Variation 6 which ranks by Structural *inLinks* only. None of these Structural re-ranking approaches

improve precision, in fact all decrease precision at 10 documents by at least 28.5% from .3082 to (at best) .2204. The results from Variation 7 (which used live WWW connectivity data) were noteworthy due to the fact that this was the only approach that increased precision over content-only results, but only from 30 to 1000 documents.

In addition to topics 401-450, we have executed a number of manual queries on the software. Some of these queries have produced quite impressive results using the WT2g dataset and associated connectivity data. See Figure 4 for an example of the results generated by a query "Vegetable Soup Recipes". The variation used to generate these results was a modification of the *RerankAdv* variation. Further research is needed to determine if these results are due to particular properties of the dataset that favours certain queries.

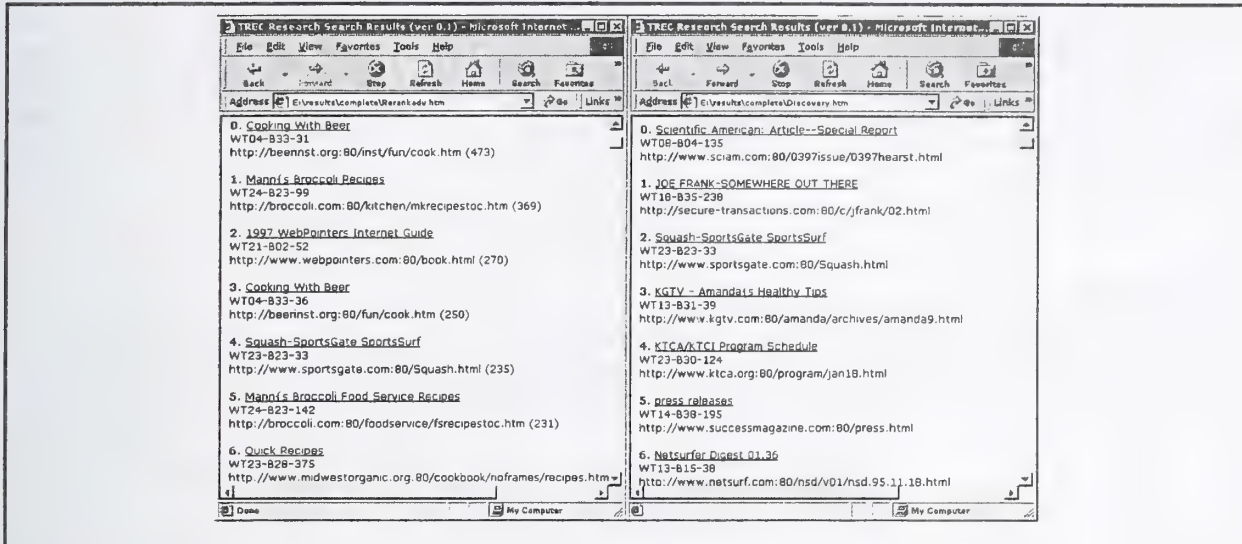


Figure 4: Example Result (left = context analysis, right = unmodified content analysis)

7. Conclusions and Future Research

We found that not enough Functional links existed in the dataset to allow many of our variations to function correctly. There was no way around this Functional link problem, except to turn to the wider WWW as a possible source of the connectivity data. See our seventh and last variation for details of how we implemented this. We are not yet in a position to draw solid conclusions from our experiments, but we plan to continue our research into this area. Of course, one of the drawbacks of these (pure) Connectivity Analysis approaches is that documents which have a large in-degree (number of *inLinks*) in the Result-Set will be ranked highly, even above other documents with a higher relevance to the query but with a smaller in-degree. This is especially true in our case, since the Result-Sets that our Content Analysis stage generated contained at most only 22.29% relevant documents. It is entirely feasible that many of our top ranked documents can not be considered relevant to the query at all, yet were ranked highly because of their popularity. We intend to implement our approaches in the future using a Confidence Factor that will indicate when and to what extent Connectivity Analysis should influence the search process. This problem of returning results that may be not entirely related to the topic represented by the query is discussed in [6,7].

Many more advanced Connectivity Analysis approaches have been developed such as those detailed in [3,6,7,10]. We didn't implement any of these approaches, as we felt that more was to be gained from developing our own ideas, with the knowledge that the other methods existed. It is our understanding that any implementation of these approaches would not succeed in improving precision (to any usable extent, if at all) when the experiments were based on the WT2g dataset, due to the lack of Functional links. We do suggest caution being taken when reviewing the Small Web Task to take the results in the context of the WT2g dataset, lest one conclude that Connectivity Analysis does not improve precision in any case. From the work of Kleinberg in [7], it is generally accepted that for queries on broad topics, Connectivity Analysis will allow for the selection of the most popular

(densely linked) documents from within a WWW community in response to a query, in addition to automatic result clustering.

Next year, we plan to partake in the Web Track using the 10GB subset and if available the 100GB collection. We hope that the 10GB dataset planned for TREC-9 will be more representative of the link structure of the WWW. If we find this not to be the case we will look into spidering our own set of HTML documents utilising a software spider that we would develop. The spider would gather documents by following the link structure of the WWW but limiting the number of documents from any particular domain (with exceptions) and implementing a priority queuing algorithm to select the most useful documents to download. It is our belief that a dataset that is gathered with the purpose of aiding experimentation into Connectivity Analysis will be vital to our future research. Our more ambitious plans for next year are supported by an increase in the amount of hardware that we have available thanks to a recent philanthropic donation to aid our connectivity research project. The hardware, based around four PIII workstations and a PIII server will be connected together on a dedicated 100Mbit/s switched network. Total storage capacity on the network is almost 200 GB and each PC is capable of expanding its number of processors as requirements dictate.

References

- [1] Li, Yanhong - **"Toward a Qualitative Search Engine"**
IEEE Internet Computing, page 24-29 (July-Aug 1998)
- [2] Amitay, Einat - **"Using Common Hypertext Links to Identify the best Phrasal Description of Target Web Documents"**
<http://www.mri.mq.edu.au/~einat/sigir/>
- [3] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson & J. Kleinberg - **"Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text"**
<http://decweb.ethz.ch/WWW7/1898/com1898.htm>
- [4] Spertus, Ellen - **"Parasite: Mining Structural Information on the Web"**
<http://atlanta.cs.nchu.edu.tw/www/PAPER206.html-admin.htm>
- [5] K. Bharat, A. Broder, M. Henzinger & P. Kumar - **"The Connectivity Server: Fast Access to Linkage Information on the Web"**
http://www.research.digital.com/SRC/personal/Andrei_Broder/cserv/386.html
- [6] K. Bharat & M. Henzinger - **"Improved Algorithms for Topic Distillation in a Hyperlinked Environment"**
ACM SIGIR'98
- [7] Kleinberg, Jon - **"Authoritative Sources in a Hyperlinked Environment"**
<http://www.cs.cornell.edu/Info/People/kleinber/kleinber.html>
- [8] S. Brin & L. Page - **"The Anatomy of a Large-Scale Hypertextual Web Search Engine"**
<http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm>
- [9] Adamic, Lada A. - **"The Small World Web"**
<http://www.parc.xerox.com/istl/groups/iea/www/smallworld.html>
- [10] S. Brin & L. Page - **"The Pagerank Citation Ranking: Bringing Order to the Web"**
<http://www-db.stanford.edu/~backrub/pageranksub.ps>

The Eurospider Retrieval System and the TREC-8 Cross-Language Track

Martin Braschler¹, Min-Yen Kan², Peter Schäuble¹, Judith L. Klavans²

¹ Eurospider Information Technology AG

Schaffhauserstrasse 18

CH-8006 Zürich

Switzerland

{braschle, schauble}@eurospider.ch

² Natural Language Processing Group

Columbia University

New York, NY 10027

USA

{min, klavans}@cs.columbia.edu

This year the Eurospider team, with help from Columbia, focused on trying different combinations of translation approaches. We investigated the use and integration of pseudo-relevance feedback, multilingual similarity thesauri and machine translation. We also looked at different ways of merging individual cross-language retrieval runs to produce multilingual result lists. We participated in both the CLIR main task and the GIRT sub task.

1. Introduction

The main aim of our participation in the cross-language track this year was to try different combinations of various individual *cross-language information retrieval* (CLIR) approaches. We reused the same corpus-based methods that we utilized last year with considerable success, while experimenting with using a number of off-the-shelf machine translation products.

We also revisited our merging approach, trying out an alternative strategy.

2. General system description

For all of our runs we used a Eurospider retrieval system, which evolved from a prototype originally created at the Swiss Federal Institute of Technology (ETH) in Zurich, with continuing development of the system now at Eurospider Information Technology AG. When indexing the different collections, we used different stemmers for the individual languages:

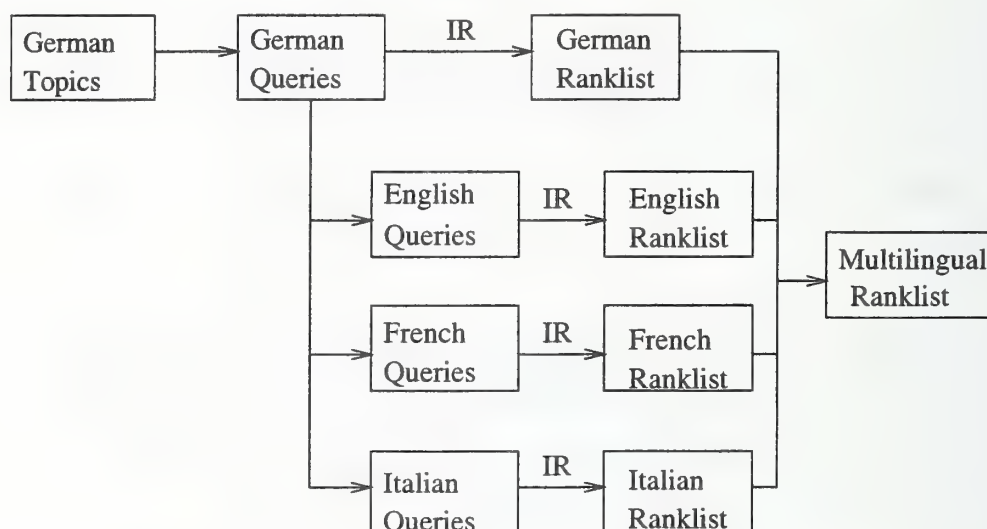
- German:
 - the stemmer distributed with the NIST PRISE retrieval system for our submissions to the main task, and;
 - the Eurospider stemmer featuring German word decomposition for the GIRT subtask submissions.
- French: the Eurospider French stemmer.

- Italian: the Eurospider Italian stemmer.
- English: the Porter English stemmer.

The retrieval status values are calculated using the *Lnu.ltn* weighting scheme as described in [6].

3. Main Task

The main task this year consisted of choosing a language in which topics are specified. Queries in that one language are then used to produce runs against all the documents in all languages (i.e. English, German, French and Italian documents). Our approach to this task is to initially produce runs using only a pair of languages, and then to merge these separate runs to produce the final, multilingual ranked list. For all submissions, we used German topics, and translated them into the other languages. This means that for all our main task submissions, we first had to obtain four runs (German monolingual, German → English, German → French and German → Italian).



The individual submissions differ in:

- the method used for query translation;
- the method used for merging of the runs, and;
- the fields of the topics used for creating the queries.

We submitted three runs for the main task, EIT99sta, EIT99mta and EIT99sal, which are explained in more detail in the following sections. All runs were automatic.

3.1. EIT99sta – Using only Automatic Compiled Resources

This run uses all three fields of the topics (title + description + narrative). It builds on methods we tested in TREC-7. The defining characteristic is that it does not use any costly, manually built linguistic resources. Instead, it uses only data structures automatically built from suitable training data.

This run uses two different methods to perform query translation: *pseudo relevance feedback* (PRF) and *similarity thesauri*. This year we used a German ↔ English similarity thesaurus, as opposed to using a manually built word list that as in TREC-7. Using the thesaurus for this run likely degraded performance, since the training data for German ↔ English was not well suited for thesaurus construction. However, we wanted to more clearly distinguish this run from run EIT99mta (see section 3.2), making this run completely free of manually constructed resources.

3.1.1. Pseudo relevance feedback

For pseudo relevance feedback, we use the fact that the TREC CLIR collections have similar content (i.e. news stories and articles), even though they are written in different languages. Therefore, we can calculate which items in these collections cover the most similar stories, using a process we call document alignment. Ultimately, we obtained three lists with pairs of the most statistically similar documents in the combined German-English collection, German-French collection and the German-Italian collection. We used these lists for the three cross-language runs. For more details on the document alignment process, see [2].

These lists are applied as follows: to retrieve French documents using a German query, we first run the German queries against the German documents, obtaining an initial result list. We then compare this list to our list of pairs of similar French and German documents. If any of the documents in the German result list have a similar French counterpart, they are replaced. If there is no matching pair, the document is discarded. Through the replacement step, we obtain a possibly shorter French result list. We then use the top documents from this list to do a pseudo relevance feedback loop, (i.e. we select the most significant terms from this set of documents using methods developed for relevance feedback — see also [3] and [5]). These terms form our French query, which we run against the French documents, to obtain a French result list.

Note that the multilingual collections used for document alignment do not necessarily have to be identical to the search collections, although they were in the case of our TREC experiments.

3.1.2. Similarity Thesaurus

A similarity thesaurus is a data structure that provides a list of terms in one language that are statistically similar to a head term in another language. Such a similarity thesaurus can be automatically built using suitable multilingual training data [4]. We built and used three such thesauri, German ↔ French, German ↔ Italian, and German ↔ English. The German ↔ French and German ↔ Italian thesauri were built on supersets of the TREC SDA data, enriched by additional years of SDA data, which was provided to us by SDA. The German ↔ English thesaurus was built using the TREC German SDA data and the TREC English AP collection.

In relevance feedback, the original query is usually expanded by terms coming from a term selection process. The new combined query is then reweighted. However, in the cross-language case, we cannot use the terms from the original query, since they are in the wrong language. We therefore replace the original query with terms selected from the similarity thesaurus. We can then apply term reweighting to combine the resulting terms from both methods, similar to the reweighting step in the classical

relevance feedback case. For this run, we added the similarity thesaurus translations only in cases when we had few documents (less than three) coming from the PRF method.

3.1.3. Runs

The four individual runs used to produce the merged result list were obtained as follows:

German monolingual: German retrieval run, followed by pseudo relevance feedback, using the top 21 ranked documents for feedback¹. This PRF loop is very similar to the multilingual case described above, only we can directly apply term selection without having to do document replacement. We used the NIST stemmer to index the documents and queries.

German → French: This run used PRF combined with a German/French similarity thesaurus built on a German/French SDA superset, as described above. We used the top 21 documents for the pseudo feedback loop.

German → Italian: This run used PRF combined with a German/Italian similarity thesaurus built on a German/Italian SDA superset, as described above. We used the top 21 documents for the pseudo feedback loop.

German → English: This run used PRF combined with a German/English similarity thesaurus built on the German SDA and English AP collections, as described above. We used the top 21 documents for the pseudo feedback loop.

3.1.4. Merging

For merging, we again used the document alignments from the pairs of the individual collections. We produced tables giving relations between scores of individual runs, making it possible to map these scores to a common range using linear regression. By repeatedly merging pairs of runs, we obtained the multilingual result lists that we submitted. This merging strategy is also detailed in [2].

3.2. EIT99mta – Adding Machine Translation Resources

As mentioned in the introduction, our aim was to test a combination of approaches to cross-language retrieval. We therefore added *machine translation* (MT) to the components used in the previous run.

This run used all topic fields, namely title + description + narrative.

3.2.1. Machine Translation

Machine translation is interesting for use in cross-language retrieval, since the majority of these systems utilize linguistic knowledge. However, we believe that MT cannot be the only solution to CLIR. This is because even though we invested considerable effort, we were not able to locate an off-the-shelf German ↔ Italian machine translation system. We think the fact that these two widely

¹ The somewhat strange number 21 is due to a minor bug. We intended to use 20 documents.

spoken European languages are not covered by commercially available software shows that very few language pairs seem to be economically viable, given the considerable effort required to build these systems². In fact, nearly all systems we were able to locate on the consumer market translate either from or to English.

3.2.2. MT Systems Used

We used the following MT systems:

German → English:

- MZ Translator from Holtschke GmbH
- T1 Translator from Langenscheidt
- Systran web translation from Systran
- Power Translator 2000 from Pons

German → French:

- MZ Translator from Holtschke GmbH
- Systran web translation from Systran

German → Italian:

- Systran web translation, using English as a pivot language (German → English → Italian)

3.2.3. Runs

For all languages, we also used the translation coming from the similarity thesaurus (see section 3.1).

We created the EIT99mta submission by combining the output from all MT systems for a given target language with the similarity thesaurus output to create an intermediate query. This query was then used in the pseudo relevance feedback loop. Our internal tests showed that using all MT systems produced small performance gains over using just the best MT system (Systran or Power Translator, depending on query fields used and query language).

3.2.4. Merging

The four individual runs were merged using the same strategy as explained in section 3.1.4.

² This does not exclude the possibility that professional systems for corporate use for this language pair exist, but these are usually priced outside of the range of many potential customers. Holtschke GmbH is advertising a German ↔ Italian system, but it uses a very small dictionary compared to their other language combinations. A few months after our TREC experiments, we became aware of a new system by LHS for German ↔ Italian. We have not been able to obtain a copy in time for this paper.

3.3. EIT99sal – Experimental Run

Our last submission for the main task used only the title + description fields of the topics. This run used a combination of pseudo relevance feedback and similarity thesaurus. Apart from the different query length, there were three modifications with respect to this run: only 10 documents were used for PRF, the similarity thesaurus translation was employed for every query, and a different merging process was used.

The merging for the first two runs, EIT99sta and EIT99mta, both calculate a relation between the retrieval status values (RSVs) of the two runs to merge. For EIT99sal, we calculated a relation between the RSVs of one run, and the *rank* of a similar document in the other run. Since RSVs tend to fall logarithmically in our system, we used logarithmic regression to obtain the relation between RSVs and ranks of the two runs.

Initial tests showed that both merging methods resulted in similar performance.

3.4. Results

The following table shows the results we obtained for the three runs we submitted for the main task.

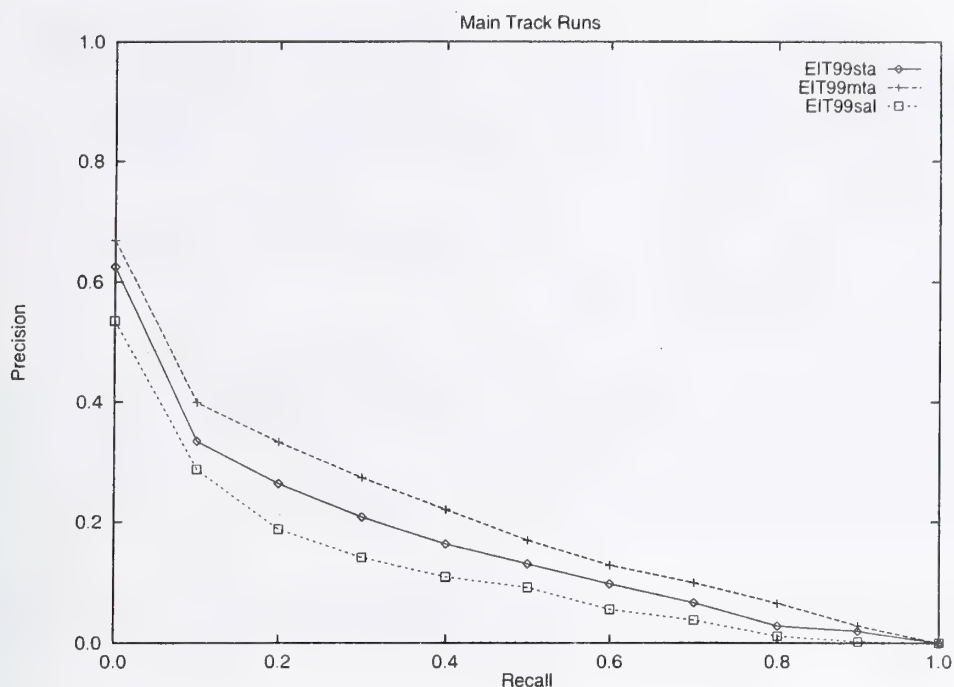
			Performance of individual queries				
Run	Avg. Prec.	R-Prec.	Best	Above	Median	Below	Worst
EIT99mta	0.1937	0.2415	2	10		16	
EIT99sta	0.1527	0.2006		9	1	14	4
EIT99sal	0.1108	0.1682	1	4	1	20	2

The results this year are somewhat mediocre, a surprise after our combination of similarity thesaurus and pseudo relevance feedback worked very well last year. We have also observed that with our new system, we don't reach the same level of performance as last year when we run the old TREC-7 queries. What exactly causes this problem is still unclear.

The rather big difference between average precision and R-precision shows that the precision seems to tail off quickly with higher recall. Some degradation may also be due to the fact that this year we used a German ↔ English similarity thesaurus, despite our belief that the training data (the German SDA in conjunction with the English AP) was not well suited for this purpose. We still need to examine these factors together with an analysis of the performance of the different merging strategies.

Not surprisingly our second run, EIT99mta, which combined most resources to produce a translation, performed best. However, for a sizeable number of queries, EIT99sta performed nearly as well, or even better (better results are obtained for 4 of 28 queries). Since this is a completely corpus-based run which did not use any manually built language resources, there are some queries which retrieve no or only very little relevant documents. These queries lower the run's performance considerably.

EIT99sal was an experimental run to examine a different method for merging. It used shorter queries, and performed poorer than the other two runs, since all the other methods we used benefited from the additional context of the longer queries.



4. GIRT Sub Task

For the GIRT subtask, the documents only exist in German, with the queries available in German, French and English. We did pure CLIR runs, ignoring both the English titles provided with the GIRT documents and the classification terms.

4.1. Runs

We submitted three runs:

- EIT99gfg, using a French ↔ German similarity thesaurus. The French queries are translated through obtaining the 20 most similar German terms from the thesaurus. No relevance feedback was used.
- EIT99geg, using an English ↔ German similarity thesaurus. Similar to the German ↔ English thesaurus, we doubted its quality due of the lack of suitable training data.
- EIT99gmt, a French ↔ German run, which used the Systran web translation to translate the queries.

All runs use all topic fields (title + description + narrative). All were automatic runs.

4.2. GIRT results

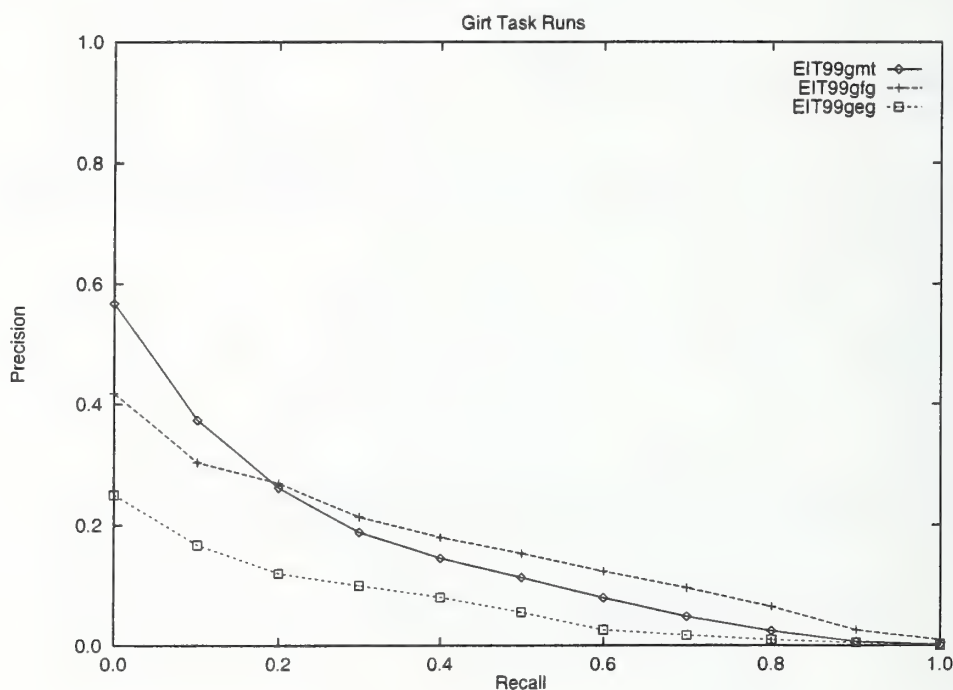
The following table shows the results we obtained for the three runs we submitted for the GIRT task.

Run	Avg. Prec.	R-Prec.	Performance of individual queries				
			Best	Above	Median	Below	Worst
EIT99gfg	0.1547	0.1844	5	4	1	14	4
EIT99gmt	0.1438	0.1965	5	3	1	15	4
EIT99geg	0.0624	0.1002	2	4		11	11

The numbers for above median and below median should be read with caution, since only few GIRT runs were submitted, making it hard to compare.

We observed however, that the results swing heavily to either side. Looking closer, we discovered that unfortunately we had a mismatch in stemming between the terms coming from our similarity thesaurus translation and the document collection. This is likely to have caused a fair number of good translated terms to be lost, which would explain the uneven performance on individual queries. We will try to analyze this problem further.

It is interesting to see that the French → German similarity thesaurus run (EIT99gfg) outperformed a high quality Systran MT run (EIT99gmt). Not surprisingly, the English → German run did much worse, again supporting our suspicion that the thesaurus is of inferior quality.



5. Thanks

Our thanks go to everyone that helped in preparing the document collections, queries and relevance assessments used in this year's CLIR track. We also thank SDA for providing us with the training data used to create the similarity thesauri.

References

- [1] M. Braschler, B. Mateev, E. Mittendorf, P. Schäuble and M. Wechsler. SPIDER Retrieval System at TREC7. In *Information Technology: The Seventh Text REtrieval Conference (TREC-7)*, pages 509-517, 1999.
- [2] M. Braschler and P. Schäuble. Multilingual Information Retrieval Based on Document Alignment Techniques. In *Second European Conference on Research and Advanced Technology for Digital Libraries*, pages 183-197, 1998.
- [3] D. K. Harman. Relevance Feedback and Other Query Modification Techniques. In *Frakes, W. B., Baeza-Yates, R.: Information Retrieval, Data Structures & Algorithms*, pages 241-261, 1992.
- [4] Y. Qiu. Automatic Query Expansion Based on A Similarity Thesaurus. *PhD Thesis, Swiss Federal Institute of Technology (ETH)*, 1995.
- [5] M. Mitra, A. Singhal and C. Buckley. Improving Automatic Query Expansion. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206-214, 1998.
- [6] A. Singhal, C. Buckley and M. Mitra. Pivoted Document Length Normalization. In *Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21-29, 1996.

TREC-8 Automatic Ad-Hoc Experiments

at Fondazione Ugo Bordoni

Claudio Carpineto and Giovanni Romano
Fondazione Ugo Bordoni, Rome Italy
{carpinet, romano}@fub.it

Abstract

We present further evidence suggesting the feasibility of using information theoretic query expansion for improving the retrieval effectiveness of automatic document ranking. Compared to our participation in TREC-7, in which we applied this technique to an ineffective initial ranking, here we show that information theoretic query expansion may be effective even when the quality of the first pass ranking is high. In TREC-8 our system has been ranked among the best systems for both automatic ad hoc and short automatic ad hoc. These results are even more interesting considering that we used single-word indexing and well known weighting schemes. We also investigate the use of term variance to refine the weighting schemes employed by our system to weight documents and queries.

the latter usually increases as the quality of the initial retrieval becomes higher, we expected that also the relative performance improvement due to query expansion would benefit from a better first pass ranking. Our results confirmed this hypothesis partially. The second goal of our participation in TREC-8 was to test the effectiveness of using the variance of term occurrence in the collection to refine the weighting schemes used to weight query and documents. The results obtained using term variance were moderately promising. Overall, the average precision of our best run in TREC-8 was 0.3106, which was a great improvement over that of our participation in TREC-7. Moreover, these results were excellent also on an absolute scale. Our system was ranked as the fourth best system for automatic short ad hoc, and as the eighth best system for automatic ad hoc. What makes these results even more interesting is that they were obtained using single-keyword indexing.

1. Introduction

This is our second participation in the TREC conference. In TREC-7 we experimented with a novel method for automatic query expansion based on an information-theoretic measure (Carpineto and Romano, 1999). The results showed that passing from unexpanded to expanded query yielded a high performance gain (+ 14%), but, on an absolute scale, the figures that we obtained were much lower than those reported by the best TREC systems (e.g., 0.1409 versus 0.3033 for the average precision). This was mainly due to the ineffectiveness of the ranking system on top of which the query expansion stage was added, which used a simple *tfidf* weighting scheme. Thus, in TREC-8 we have tried to improve the basic ranking system in the hope of increasing not only the retrieval effectiveness of the document ranking produced in response to a query (whether unexpanded or expanded) but also the utility of the query expansion itself. Since

2. Test collection indexing

1. *Text segmentation.* Our system first identified the individual terms occurring in the test collection, ignoring punctuation and case.

2. *Word stemming.* To extract word-stem forms, we used a very large *trie*-structured morphological lexicon for English (Karp et al, 1992), that contains the standard inflections for nouns (singular, plural, singular genitive, plural genitive), verbs (infinitive, third person singular, past tense, past participle, progressive form), adjectives (base, comparative, superlative).

3. *Stop wording.* We used a stop list, contained in the CACM dataset, to delete from the texts common function words. In addition, for efficiency reasons, we removed the terms that appeared in more than 75000 and less than 5 documents.

All the test collection indexing was of the single-keyword type. In particular, we used no manually-

predefined multiword phrases to conflate groups of related words into single concepts.

3. First pass ranking

We used Okapi formula (Robertson et al., 1999) for matching queries and documents in the first pass ranking:

$$sim(q, d) = \sum_{t \in q \cap d} w_{d,t} \cdot w_{q,t} \quad (1)$$

with $w_{d,t}$ given by

$$\frac{(k_1 + 1) \cdot f_{d,t}}{k_1 \cdot \left[(1-b) + b \cdot \frac{W_d}{avr_W_d} \right] + f_{d,t}} \quad (2)$$

and $w_{q,t}$ given by

$$\frac{(k_3 + 1) \cdot f_{q,t}}{k_3 + f_{q,t}} \cdot \log \frac{N - f_t + 0.5}{f_t + 0.5} \quad (3)$$

where k_1 , k_3 and b are constants which were set to 1.2, 1000, and 0.75 respectively. W_d is the length of document d expressed in words and avr_W_d is the average document length in the entire collection. The value N is the total number of documents in the collection, f_t is the number of documents in which term t occurs, and $f_{x,t}$ is the frequency of term t in either document d or query q .

4. Information-theoretic query expansion

To automatically expand the query we used the Rocchio formula (Rocchio, 1971) coupled with an information-theoretic term scoring function, similar to the approach described in (Carpineto et al., 1999). The Rocchio formula for pseudo-relevance feedback is:

$$Q_{new} = \alpha \cdot Q_{orig} + \frac{\beta}{|R|} \sum_{r \in R} r \quad (4)$$

where Q_{new} is a weighted term vector for the expanded query, Q_{orig} is a weighted term vector for the original unexpanded query, R is a set of top retrieved documents (assumed to be relevant), and r is a weighted term vector extracted from R .

Using basic Rocchio, the weights of the terms contained

in Q_{orig} and r are determined considering their primary weights, i.e., the weight of each term as determined by the weighting scheme used to produce the first pass ranking. In our approach, Q_{orig} was indeed the weighted query vector used in the first pass ranking, i.e., as determined by expression (3), but then we used a different method than expression (2) to weight each expansion term contained in r . Our weighting was based on the Kullback-Lieber distance between the distribution of the term in R and the distribution of the term in the whole collection. More precisely, each expansion term was assigned a score given by:

$$score(t) = [p_R(t) - p_C(t)] \cdot \log [p_R(t) / p_C(t)] \quad (5)$$

where $p_X(t)$ is the probability of occurrence of term t in the set of documents X , R indicates the pseudo-relevant set, C indicates the whole collection. The main rationale of using a term-scoring function based on distribution analysis to reweight expansion terms is that in this way a term that is good for a given query can receive a high score even when its weight in the collection is low, whereas with basic Rocchio there may be a mismatching between the relevance of a term to a given query and the weight actually assigned to it (Carpineto and Romano, 1999).

From a practical point of view, we considered as expansion candidates all terms contained in R , and then selected those with the highest score. To estimate $p_X(t)$, we used the ratio between the number of occurrences of t in X , treated as a long string, and the total number of terms in X . The other parameters involved in this method were chosen as follows. The constant α and β in expression (1) were set to 1 and 1.5, respectively; we used 12 pseudo-relevant documents and 50 expansion terms were considered for inclusion in the expanded query. The choice for the values of these parameters was based on earlier results obtained in past TREC conferences and on some experiments that we performed on the TREC-7 data.

5. Refining query and document weighting with term variance

Classical weighting scheme usually do not take into account the variance (σ^2) of term occurrence in the collection of documents. This may be a useful information to identify terms that truly differentiate and relate subsets of documents. Generally speaking, the lower the variance of a term, the less likely it is that the term is a good one. Common function words, for

instance, are likely to exhibit a low variance because they tend to occur in similar proportions in all documents. In order to compute the variance of a term, we considered the whole set of documents (including those that did not contain that term) and used two alternative methods to compute the term occurrence in a document: term frequency (tf), and normalized term frequency wrt document length (norm-tf).

We used the term variance to modify both the document weights and the scores used to weight the expansion terms. To change the document weights, we simply multiplied expression (2) by $\log \sigma^2$. The modification of the expansion terms weights required more caution, because the introduction of σ^2 may interact with the information-theoretic measure used to weight the expansion terms in the first place. Recall that, using expression (5), a term is assigned a high score if it is much more concentrated in the top documents than the whole collection. This indication about the goodness of a term may be inversely related to the term variance, in the sense that a small variance may further imply that the term appears only in the top documents while a high variance may suggest that the behavior captured by expression (5) will be more likely attributed to chance. This implies that the terms to be favored to refine query weighting are those with small variance, rather than high variance. In fact, experimenting with TREC-7, we noticed that using σ^2 as a factor hurt performance, while using its inverse improved performance. Thus, we modified the weights of expansion terms by the inverse of $\log \sigma^2$.

It is useful to examine the relation between σ^2 and the *idf* factor ($\log N/df$), because the latter is a relative frequency measure of the same kind as σ^2 , with a similar goal: identifying terms that help distinguish the documents to which they are assigned from the remainder of the collection. One might think that σ^2 is directly related to *idf* (e.g., the higher *idf*, the higher σ^2), but it turns out that this is not the case. A high value of *idf* will usually produce a low value of σ^2 , while a low value of *idf* may be associated to a high as well as to a low value of σ^2 , depending on the distribution of terms in the documents in which they are contained. On the other hand, if we considered only the documents that contain the given term, the value of σ^2 would be totally independent of the value of *idf*, irrespective of how we chose to compute the term occurrence. In fact, σ^2 and *idf* are based on different variables (number of occurrences versus binary value of occurrence/non-occurrence) and perform different statistical operations on those variables (variance versus mean). Therefore they seem to capture different patterns of data regularities and can be used together in a comprehensive weighting scheme.

6. Results

In Table 1 we show the performance of four different document rankings with expanded query. The four runs were characterized by the following parameters: title + description, without variance; title + description + narrative, without variance; title + description + narrative, with σ^2_{tf} ; title + description + narrative, with $\sigma^2_{norm-tf}$. In Table 1 we also show the results of document ranking with unexpanded query used as a baseline. The results are reported using the standard TREC performance evaluation measures.

Table 1 shows that the four rankings with expanded query had better results than unexpanded query for virtually all evaluation measures, including precision for the first retrieved documents. Of the four expanded runs, those using T+D+N fared consistently better than the one with only T+D, with small, but not negligible, differences. The introduction of variance in the method employing the full topic description was beneficial, especially as far as average precision was concerned. In particular, we obtained the best average precision result – i.e., 0.3106 – using $\sigma^2_{norm-tf}$.

As mentioned before, we wanted to test the hypothesis that when using a better baseline retrieval the shift from unexpanded query to expanded query would increase the relative performance improvement. While the performance of ranking with unexpanded query in TREC-8 was higher than double what it was in TREC-7, the relative performance improvement after query expansion in TREC-8 was the same as TREC-7 (+14%). Thus, the relative performance improvement due to expansion was not as high as expected. One possible explanation for this is that when the initial retrieval is really good, it can be hardly improved further upon. The reported results were obtained by averaging over the whole set of topics; a topic by topic analysis might help better understand when and why the relative performance variations are different. The second main goal of our experiments was to test the effectiveness of the use of term variance in the weighting scheme. Our results provide some evidence that it may be a promising directions, but of course this issue needs to be investigated more carefully.

It is also useful to compare the overall performance of our system with that of the other official runs in the Ad-hoc category.

Considering average precision as the measure for performance comparison, our best runs for automatic ad hoc (fub99tt) and for automatic short ad hoc (fub99td) were ranked as the eight and the fourth best system, respectively. A query by query analysis revealed that we achieved better than median performance for 37 topics in automatic ad hoc and for 40 topics in automatic short

ad hoc. In automatic ad hoc we obtained the best performance results for two topics, in automatic short ad hoc for five topics. Finally, it should be noted that the documents retrieved by our best runs could not be

included in the document pool used to produce the topic relevance file by the TREC's assessors. Thus, their performance might have been better than actually reported.

Table 1. Comparative performance of ranking with and without query expansion

	unexp. T+D+N	exp. T+D	exp. T+D+N	exp. T+D+N, σ^2_{tf}	exp. T+D+N, $\sigma^2_{norm-tf}$
Run tag		fub99td	fub99a	fub99tf	fub99tt
Ret&Rel	2938	3298	3262	3281	3299
AV Prec	0.2718	0.3064	0.3068	0.3099	0.3106
11 Point Prec	0.2978	0.3229	0.3245	0.3281	0.3285
R-Prec	0.3168	0.3366	0.3364	0.3398	0.3354
Prec at 5	0.5960	0.5760	0.6080	0.6040	0.6160
Prec at 10	0.4920	0.5100	0.5300	0.5260	0.5300

7. Conclusion

Our experiments show that information-theoretic query expansion may produce excellent results, both on a relative and on an absolute scale. In addition, they seem to imply that the relative performance improvement due to query expansion does not grow monotonically as the quality of the initial baseline retrieval improves. Finally, they also suggest that it may be useful to investigate the use of term variance to refine the weighting schemes employed to weight documents and queries.

Acknowledgments

This work has been carried out within the framework of an agreement between the Italian PT Administration and the Fondazione Ugo Bordoni.

References

Carpineto, C., De Mori, R., and Romano, G. (1999). Informative term selection for automatic query

expansion. *Proceedings of the Seventh Text Retrieval Conference (TREC-7)*, pp.363-369, NIST Special Publication 500-242.

Carpineto, C., and Romano, G. (1999). Towards better techniques for automatic query expansion. *Proceedings of the 3th European Conference on Digital Libraries (ECDL'99)*, Paris, France, pp. 126-141, Lecture Notes in Computer Science 1696, Springer.

Karp, D., Schabes, Y., Zaidel, and M., Egedi, D. (1992). A freely available wide coverage morphological analyzer for English. *Proceedings of the 14th International Conference on Computational Linguistics (COLING '92)*, Nantes, France.

Robertson, S. E., Walker, S., and Beaulieu, M. (1999) Okapi at TREC-7: Automatic ad hoc, filtering, VLC, and interactive track. *Proceedings of the seventh Text REtrieval Conference (TREC-7)*, pp. 253-264, NIST Special Publication 500-242.

Rocchio, J. (1971). Relevance feedback in information retrieval. In Salton, G. (ed.), *The SMART retrieval system - experiments in automatic document processing*, chapter 14, Prentice Hall, Englewood Cliffs.

Natural Language Information Retrieval: TREC-8 Report

Tomek Strzalkowski

GE Research & Development

Jose Perez-Carballo

Rutgers University

Jussi Karlgren

Swedish Institute of Computer Science

Anette Hulth

Stockholm university

Pasi Tapanainen & Timo Lahtinen

Conexor OY, Helsinki

1. Summary

This report describes the adhoc experiments performed by the GE/Rutgers/SICS/SU/Conexor team in the context of TREC-8. The research efforts went in four directions:

1. As in previous years, we performed a full linguistic analysis of the entire corpus, and used the results of the analysis to provide index terms on a higher level of abstraction than can be provided by stems alone.
2. We made use of two different query expansion techniques, one automatic and one manual, both developed for TREC-8.
3. The various analysis models were combined using a stream model architecture, where each stream represents an alternative text indexing method, and the stream's various overlapping knowledge was merged using a new merging algorithm derived from first principles.
4. The entire text was analyzed for various stylistic items.

Due to the distributed approach, this years' research efforts partly canceled out each other. New experiments in every step of the process did not result in an overwhelming overall result. We are able to determine that the manual query expansion technique developed at General Electric performed very well.

2. Background

The work reported here was part of the Natural Language Information Retrieval project (NLIR) (Perez-Carballo, Strzalkowski, 1999; Strzalkowski et al., 1998 and 1997; Strzalkowski, 1995). One of the thrusts of this project has been to demonstrate that robust NLP techniques can help to derive better representation of text documents for indexing and search purposes than any simple word and string-based methods commonly used in statistical full-text retrieval. This was based on the premise that linguistic processing can uncover certain critical semantic aspects of document content, something that simple word counting cannot do, thus leading to a more accurate representation. In earlier experiments we demonstrated that NLP could be done efficiently on a very large scale, and that it could have a significant impact on the performance of the IR systems we were using then. At the same time, it became clear

that exploiting the full potential of linguistic processing is harder than originally anticipated. In particular, simple linguistically motivated indexing (LMI) techniques turned out to be no more effective than well-executed statistical approaches, at least for English, while more advanced NLP techniques, such as concept extraction, remained too expensive for large-scale applications (Sparck-Jones, 1999).

Given this state of affairs, we went on to investigate specific conditions under which LMI could be more beneficial. For example, we have noticed that the amount of improvement in recall and precision which we could attribute to NLP, appeared to be related to the type and length of the initial search request. Longer, more detailed topic statements responded well to LMI, while terse one-sentence search directives showed little improvement. This is not particularly surprising considering that the shorter queries either contain a handful of highly discriminating terms or are deliberately vague. We adopted the topic expansion approach in which the original topic is expanded using passages selected from sample retrieved documents. The intent was to expand the initial search specifications in order to cover their various angles, aspects and contexts. Based on the observations that NLP is more effective with highly descriptive queries, we designed an expansion method in which passages from related, though not necessarily relevant documents were imported into the user queries. This method produced a fairly dramatic improvement in the performance of several different statistical search engines that we tested boosting the average precision by anywhere from 40% to as much as 130%. Therefore, we concluded that topic expansion appears to lead to a genuine, sustainable advance in IR effectiveness. Moreover, we showed in TREC-7 that this process can be automated while maintaining at least some of performance gains. Thus far we have used only very simple linguistic tools (i.e., those suitable for high-volume IR applications) to assist automatic expansion, but we see this area as ripe for more advanced processing techniques, including entity and event extraction, co-reference and cross-reference techniques, etc.

3. Processing scheme

InQuery was used as the indexing and retrieval engine. This year, the linguistic processing of TREC data, both text and queries, was performed in Helsinki using the newly developed Functional Dependency Grammar (FDG) text processing toolkit. The processed text was sent via ftp to Rutgers and SICS for indexing.

For some of the manual submissions, the topics were processed at General Electric using the interactive Query Expansion Tool for manual query expansion; for the automatic submissions, queries were expanded at Rutgers using a passage retrieval algorithm. The expanded topics were processed in Helsinki to obtain matching search terms for the linguistic indices, and sent back to Rutgers for retrieval.

The results from the various processing and retrieval streams were merged to obtain a final rank order using a merging algorithm developed for this year's TREC at Stockholm university and SICS.

4. Streams

The stream model (see Fig. 1) has been described in previous TREC papers and in (Perez-Carballo, Strzalkowski, 1999). Each "stream" uses its own index which is created using a different indexing technique (some of them involving linguistic processing). The results obtained from all the streams are combined using merging algorithms. In past TRECs we were able to obtain significant improvements in performance over the baseline single word indexing stream. The experiments for TREC-8 did not yield similar improvements. This year, in spite of improved analysis machinery, the linguistic streams performed far less well than earlier years and we were not able to combine them usefully with the standard

retrieval streams. There are several possible explanations, but the main reason appears to be the several simultaneous changes in our approach. We changed the character of the streams, reworked the merging algorithm, and as a result we have not been able to make use of previous years' experience in matching query processing with text processing and combining results appropriately in time for this report. We are continuing the experiments that we expect to publish elsewhere.

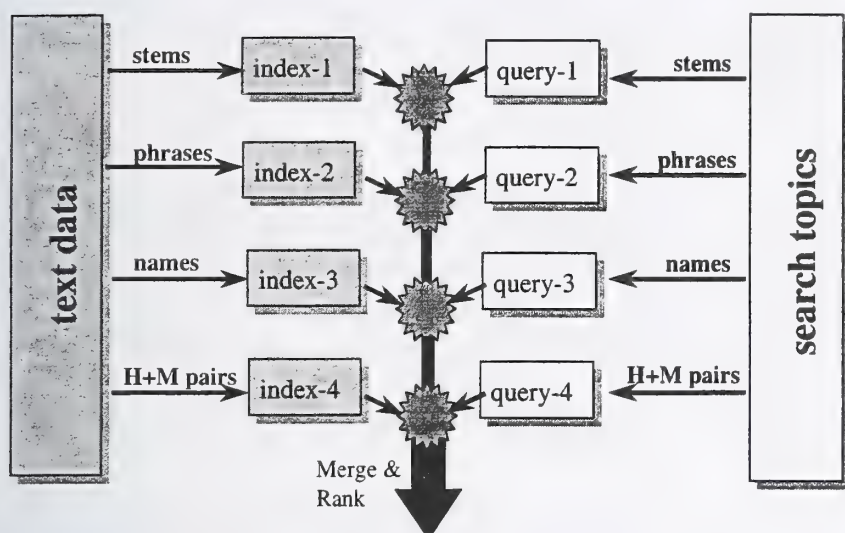


Figure 1. Stream Model organization

4.1 Linguistic Streams

Some of the linguistic streams we used were created using Helsinki's Functional Dependency Grammar (FDG) which includes the EngCG-2 tagger and dependency syntax which links phrase heads to their modifiers and verbs to their complements and adjuncts. FDG was applied to the whole corpus, with the output passed to the stream extractor.

We tried to merge the results obtained from linguistic streams with the stems stream, as we have done other years, but were unable to obtain good results (i.e. improve the performance of the stem stream). Because of lack of disk space we could not use the same automatic expansion algorithm on the linguistic streams so we cannot draw any conclusions yet about linguistic streams or the merging algorithms.

In one experiment we used the InQuery #phrase (see below) operator in order to add phrases from one of the linguistic streams to the query generated using manually chosen summaries. This seemed to actually decrease the performance of that run.

In order to have a baseline to be used with the linguistic streams we added InQuery's #phrase operator to words that appeared close to each other in the topics. No linguistic processing was used at all. This was done automatically. Some of the "phrases" obtained did not seem to make any sense and no human would have added them. Surprisingly, the queries that used that device performed better than the lin-

guistic streams and in some cases better than the pure stems (some of our "official" runs, reported below, use this technique). We are now performing further experiments and tests. More definitive results and analysis will be presented in the final form of this paper.

InQuery's #phrase operator (quoting from the InQuery manual):

Phrase Operator: #phrase(T1 ... Tn)

Terms within this operator are evaluated to determine if they occur together frequently in the collection. If they do, the operator is treated as an ordered distance operator of 3 (#od3). If the arguments are not found to co-occur in the database, the phrase operator is turned into a SUM operator. In ambiguous cases the phrase becomes the MAX of the SUM and the OD3 operators.

5. Merging the streams

In previous years, the merging algorithm for TREC was tuned through trial and error, and we always managed to find relative weighting that improved the score. This process is of course unsatisfactory and we tried to capture some dependencies that would help to automatically estimate the merging function. For the past two TRECs we added a rank dependent non-linear weighting scheme in which a document's final rank in the merged ranked list is a function of average precision of component streams, as well as of the rank of the document rank in the various streams. This change had a positive effect on merging precision, but it still required supervised training in order to optimize the parameters.

This year, we decided to use a more principled approach. We performed a set of merging experiments using some streams that we judged the most promising based on early experiments. For lack of time and processing space, and trivial file transfer problems, we restricted the experiments to the following four streams, postponing the inclusion of much of the linguistic and stylistic experiments made during the course of the project:

1. run.7.proc.PH.t3d1n1.35: automatic expansion, proximity phrases, words from title are repeated 3 times, runs on *stem* stream.
2. run.7.ph.t3d1n1.35: before expansion the terms from the *ph* stream are added to the topics using the #phrase operator in case it is a phrase, automatic expansion, words from title are repeated 3 times, runs on *stem* stream.
3. run.7.proc.P.t3d1n1.35: automatic expansion, words from title are repeated 3 times, runs on *stem* stream.
4. run.7.thr.t3d1n1: words from title are repeated 3 times, runs on the *thr* stream.

Training several different classifiers and combining the predictions of these into a single prediction is a common method for creating an accurate classifier from a set of training data (Breiman, 1996; Drucker, et al, 1994; Wolpert, 1992). A number of researchers have demonstrated that such combined classifiers in general are more accurate than any of the constituent classifiers (Dietterich, 1997; Breiman, 1996; Merz, 1999; Quinlan, 1996; Wolpert, 1992; Zhang, Mesirov & Waltz, 1992). Bartell, Cottrell and Belew (1994) have shown similar results in the document retrieval domain: using different retrieval algorithms and then combining them may significantly improve retrieval performance.

The streams in our model capture different aspects of the documents' content. When merging the streams, the aim is to produce a final result that is more accurate, i.e., has a higher average precision,

than the output of any of the individual streams. The final result should therefore be a richer set of documents. Obtaining this result is, however, not trivial.

As detailed above we compose a mixture of different indexing approaches, term extracting, weighting strategies, and different search engines we use into indexing streams. Each stream represents an alternative text indexing method; some require complex linguistic processing, while others are based on simple quantitative techniques. The results obtained from the different streams are lists of documents ranked in order of relevance: the higher the rank of a retrieved document, the more relevant it is presumed to be (in comparison to the other retrieved documents). The ordering is based on the relevance score - a figure produced by the stream, reflecting the document's accuracy as judged by the system. The streams perform in parallel and the results from the different streams should be merged to produce one final result. As the streams capture different aspects of the documents' content, the final result should be a richer set of documents. The aim of the merging is to produce a final result that is more accurate than the output of any of the individual classifiers. Obtaining such a result is, however, not trivial.

A merging algorithm called SEQUEL (Asker & Maclin, 1997) was implemented for the task. The rationale behind SEQUEL is to find the most confident classifier down to a certain threshold. It requires that the lists are sorted by - in this case - relevance score. The confidence is calculated by finding the classifier with the highest proportion correct classifications (i.e., relevant documents) at the top, down to the first non-relevant one. The threshold will be the lowest relevance score within this interval. The items covered by the span are removed from all classifiers. At a certain value the best performing classifier is considered the default classifier, i.e., it is used as the single classifier.

The algorithm was trained on 40 out of 50 queries – setting aside 10 queries for testing – from the TREC-7 materials, using the TREC-7 data and relevance judgments. All non-judged documents were removed, leaving only the judged documents for consideration. Two different implementations of the algorithm were made: one where all queries were sorted by the judgment of the system; and one where the program examined the confidence for each query at the time, taking the average as the result.

Although the algorithm performed well, the combined classifier did not beat the best individual classifier. This could possibly be because of the algorithm not being very “forgiving”: immediately upon finding an irrelevant document the stream is discarded. SEQUEL also tended to work with “chunks” of documents, covering too many at the time. This could be due to the fact that the relevance scores given by the retrieval systems range over a quite limited span. (An implementation that normalized the ranking scores to fall between 1 and 0 was also made, but the improvement was not significant.)

The algorithm tended to favor the best performing classifier (it being the most confident stream) and discard the additional information that the weaker streams may contribute with. Neither does the method take a possible overlap of retrieved documents in different streams into account. The algorithm was implemented to consider the top 1000 for each question and classifier. This means that for the majority of the documents, we only had judgments from one or two streams. It would be more appealing to apply a method where every document could take advantage of the fact that we use several different retrieval methods.

6. Automatic expansion algorithm description.

Using the automatic expansion algorithm described in this section we obtained a 37% improvement of average precision over a baseline where no expansion was used.

6.1 Algorithm:

1. The topics sent by NIST were processed to eliminate some words and phrases such as: "a relevant document".
2. The title text was repeated 3 times, the description 2 times. Our intent was to give different weights to the different fields.
3. Processed topics were submitted to InQuery in order to retrieve the top 20 documents.
4. For each one of the D documents with highest document score larger than threshold T, extract all passages of size larger than S. Let the passage score be the sum of all unique occurrences of a query term (either word or phrase) in that passage.
5. Choose the P passages with highest passage score and add them to the original query. Notice that given two passages A and B, the score of B may be higher than A (and thus B may be chosen over A) even though A may belong to a document that has a higher score than B's document.

The values used for the parameters described above were: D = 5; T = .432; S = 50; P = 12

7. Ad-Hoc submissions

We submitted 4 runs for the ad-hoc track.

query id	Relevant documents retrieved out of 4728	Average precision	Precision at 10 documents	R-Precision	query description
1. 8manexT3D1N0 (judged)	0.3325	0.3346	0.6520	0.3671	manually-assisted topic expansion using only title and description fields.
2. GE8ATDN1 (judged)	0.3138	0.2623	0.5020	0.2984	automatic topic expansion using title description and narrative fields. Proximity phrases only.
3. GE8ATDN2 (not judged)	0.3068	0.2580	0.5498	0.2993	automatic topic expansion using title description and narrative fields. No phrases used.
4. GE8ATD3 (not judged)	0.3022	0.2618	0.5658	0.2959	automatic topic expansion using only title and description fields. Proximity phrases plus original text.

Below are short descriptions of our official ad-hoc runs (shown in the table above). All official runs submitted were produced using only the stem stream as opposed to being the result of merging evidence from different streams.

7.1 Summarization-based manually-assisted topic expansion run (8manexT3D1N0)

Manually-assisted topic expansion using only title and description fields. The methods used to obtain this run were the same as the ones we used in TREC-7. Summaries used in expansion were derived from top-ranked documents retrieved by SMART using the initial topics (title+description only). The key characteristics of this run is the 10 minute time limit imposed on topic expansion. All expansion has been performed via the Query Expansion Tool interface (QET) which allows the user to view only the summaries of top retrieved documents, and select or deselect them for topic expansion. By default, summaries of all top 30 documents were used for expansion unless the user manually deselected some (this was precisely the only form of manual intervention allowed.) We observed that for many queries 2 interactions were possible within the 10 minute interval. The first interaction (submit original query, wait for result, get 30 summaries, review & deselect summaries, and commit the selections) would take typically 4-6 minutes. In the second interactions, only the new documents retrieved in top 30 ranks (if any) were considered, therefore usually 3-4 minutes were sufficient. The target of expansion was to get between 5 and 10 “relevant” summaries within the allotted time. If this was achieved within the first interaction, no further search was performed. Otherwise, the second interaction was attempted if at least 3 minutes remained. This 6-4 split was determined in dry-run trials with TREC-6 queries. The topic expansion interaction proceeds as follows:

1. The initial natural language topic statement is submitted to a standard retrieval engine via a Query Expansion Tool (QET) interface. The statement is converted into an internal search query and run against the database.
2. The system returns topic-related summaries of top N (=30) documents that match the search query.
3. The user reviews the summaries (approx. 5-15 seconds per summary) and de-selects these that are not relevant. For TREC-8 evaluations (like for TREC-7), we set a time limit of 10 minutes per query (clock time).
4. All remaining summaries are automatically attached to the original search topic.
5. The expanded topic is passed through a series of natural language indexing steps and then submitted for the final retrieval.

7.2 Automatic topic expansion run (GE8ATDN1)

This run uses title description and narrative fields. Proximity phrases only: the query text is replaced by the output of an algorithm that linked words which appeared in the same sentence at less than 3 words of each other using the #phrase operator. This run was intended as a baseline but became an official run when we were unable to beat its performance.

7.3 Automatic topic expansion run (GE8ATDN2)

This run uses title description and narrative fields, like GE8ATDN1, but no phrases are used.

7.4 Automatic topic expansion run (GE8ATD3)

This run, again, uses only title and description fields. It uses “proximity phrases” formed out of words occurring together.

8. Continuing work

We are currently performing a set of new experiments that takes into account the rankings obtained from every stream. All documents that have been ranked among the first 1000 documents by at least

one stream out of N streams constitute a “document pool” of at least 1000 documents and not more than $N \cdot 1000$ documents. We let every stream score all documents in this pool. As a first experiment, we will implement a simple linear combination of the judgments from the four streams. A weighted sum of all the scores from each of the streams will get us the total score that includes the knowledge of all streams in question. As mentioned before, the span for the ranking scores is not that large, and therefore even a very small score can alter the ordering of the documents. For these tests we will retain the non-judged documents.

There is a possibility that some documents could get a better total score by having been given four low scores (too low to be among the best 1000 documents for any stream) than one with a high score on one and no rating on the other streams. However, if none of our streams ranks a document among the top 1000 we will discard it. If the experiments with simple linear combinations turn out satisfactory – i.e. better than the non-adapted learning algorithms – we will continue with more sophisticated methods, by for example weighting the different streams.

The main point with more “forgiving” classifiers, is to not discard a stream immediately upon finding an irrelevant document: document relevance is a debatable issue in itself, and cannot easily be compared to other classification tasks where the errors are of a more clear-cut nature.

8.1 Further experiments

This paper is just a preliminary report. Before we present a final version for the TREC-8 proceedings we must complete at least the following experiments:

- Create indexes for all linguistic streams that allow for query expansion using the same algorithm used for the *stem* stream. Try merging algorithms again but with runs obtained using query expansion.
- Merge: *stem* run (without expansion) with some (or each of the) linguistic stream(s) (without expansion). Can we get any improvement from merging? If yes, then expansion is drowning the improvement obtained by linguistic streams.
- Compare performance of automatic expansion using passages and automatic expansion using summaries.
- Compare the performance of runs obtained using LMI vs. runs obtained linguistic processing on the queries only.

9. Conclusions

Preliminary results seem to suggest it would be possible to get as good a performance by processing the query (including expansion) and using an IR system with an expressive query language such as InQuery as what we get creating indexes using sophisticated and very expensive linguistic techniques. We should explore the possibilities of using a much more sophisticated linguistic analysis of the queries but less on the index.

10. References

Asker, L. & Maclin, R. (1997). Ensembles as a Sequence of Classifiers. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97.

- Bartell, B.T., Cottrell, G.W. & Belew, R.K. (1994). Automatic Combination of Multiple Ranked Retrieval Systems. In: Proceedings of the seventeenth annual international ACM-SIGIR conference on Research and development in information retrieval.
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, vol. 24(2) pp. 123-140.
- Callan, Jamie. 1994. "Passage-Level Evidence in Document Retrieval." *Proceedings of ACM SIGIR'94*. pp. 302-310.
- Clemen, R. (1989). Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, vol. 5 pp. 559-583.
- Drucker, H., Cortes, C., Jackel, L., LeCun, Y., and Vapnik, V. (1994). Boosting and other machine learning algorithms. In: *Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann.
- Jarvinen, T., and Tapanainen, P. 1997. "A dependency parser for English." Tech. Rep. TR-1, Department of General Linguistics, University of Helsinki, Finland.
- Jarvinen, T., and Tapanainen, P. 1998. "Towards an implementable dependency grammar." In *Processing of Dependency-Based Grammars*. Montreal, Canada. S. Kahane and A. Polguere, Eds., COLING-ACL'98, Association for Computational Linguistics, Universite de Montreal, pp. 1-10.
- Kwok, K.L., L. Papadopoulos and Kathy Y.Y. Kwan. 1993. "Retrieval Experiments with a Large Collection using PIRCS." *Proceedings of TREC-1 conference*, NIST special publication 500-207, pp. 153-172.
- Merz, C. J. (1999). Using Correspondence Analysis to Combine Classifiers. *Machine Learning*, vol. 36(1/2) pp. 33-58.
- Perez-Carballo, Jose, Strzalkowski, Tomek (1999) Natural language information retrieval: progress report. *Information Processing and Management*, Vol. 36, No. 1, pp. 155-178. Pergamon/Elsevier.
- Quinlan, J. R. (1996). Bagging, boosting, and c4.5. In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.
- Sparck-Jones, Karen. 1999. "What Is The Role for NLP in Text Retrieval". In T. Strzalkowski (ed.) *Natural Language Information Retrieval*. Kluwer. pp. 1-25.
- Strzalkowski, T., Stein, G., Wise, G.B., Perez-Carballo, J., Tapanainen, P., Jarvinen, T., Voutilainen, A. & Karlgren, J. (1998). Natural Language Information Retrieval: TREC-7 Report. In: *Proceedings of the Seventh Text REtrieval Conference (TREC 7)*. NIST Special Publication, Gaithersburg: NIST.
- Strzalkowski, Tomek, Fang Lin, Jose Perez-Carballo, and Jin Wang. 1997. "Natural Language Information Retrieval: TREC-6 Report." *Proceedings of TREC-6 conference*.

- Strzalkowski, Tomek, Louise Guthrie, Jussi Karlgren, Jim Leistsnider, Fang Lin, Jose Perez-Carballo, Troy Straszheim, Jin Wang, and Jon Wilding. 1997. "Natural Language Information Retrieval: TREC-5 Report." Proceedings of TREC-5 conference.
- Tapanainen P & Järvinen T, 1997, A non-projective dependency parser. ANLP'97, p. 64-71, Washington DC.
- Tapanainen P., 1999, Parsing in two frameworks: finite-state and functional dependency grammar. Ph.D. thesis, Language technology, University of Helsinki .

Ad hoc, Cross-language and Spoken Document Information Retrieval at IBM

Martin Franz, J. Scott McCarley, R. Todd Ward
IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
<franzm, jsmc, tward>@watson.ibm.com

1 Introduction

The Natural Language Systems group at IBM participated in three tracks at TREC-8: ad hoc, SDR and cross-language. Our SDR and ad hoc participation included experiments involving query expansion and clustering-induced document reranking. Our CLIR participation involved both the French and English queries and included experiments with the merging strategy.

2 Ad Hoc Track

In the TREC-8 ad hoc experiments we used a two-pass approach, in which the top documents, as ranked by the Okapi formula [1], were used to construct expanded queries, which were then used to compute the final scores. We also experimented with applying a clustering algorithm to obtain a more reliable list of passages for query expansion.

The data pre-processing algorithm was similar to the one we used in our previous TREC participations [2], [3]. It consisted of a decision tree based tokenizer, part-of-speech tagger [4] and a morphological analyzer. Filler query prefixes were removed using a database of such prefixes from previous TREC query sets. Morphed document and query unigrams and bigrams were collected using a vocabulary of 540459 words and a stop list of 514 words.

We applied the Okapi formula [1] in the first pass scoring the as described in [3]. First pass results are summarized in Table 1, line 1. Based on the first pass passage ranking, we constructed expanded queries using an LCA technique [5], modified as described in [3]. Both the documents and passages were scored with respect to the expanded queries using the Okapi formula. The final (pass 2) score of a document was computed as a combination of the document's score

	title		description		title + description	
	AveP	P20	AveP	P20	AveP	P20
pass1	0.2480	0.4010	0.2241	0.3760	0.2613	0.4270
pass2	0.2784	0.4090	0.2531	0.3950	0.3005	0.4500

Table 1: Ad hoc retrieval results, automatic.

	passage		document	
	AveP	P20	AveP	P20
baseline TREC-7	0.2032	0.3490	0.2140	0.3820
clustering TREC-7	0.2091	0.3630	0.2154	0.3920
baseline TREC-8	0.2480	0.3980	0.2481	0.3970
clustering TREC-8	0.2507	0.3910	0.2491	0.3950

Table 2: The effect of document clustering on selecting passages for query expansion, description query fields.

and the score of its highest ranking passage. Second pass results are shown in Table 1, line 2.

We also experimented with a clustering algorithm used to augment the list of passages used for query expansion, attempting to reduce the influence of the passages that rank high in the first pass scoring, but have little in common with the rest of the high ranking passages. In this experiment, we clustered the list of top 1000 passages from the first pass using a technique described in [7] and [8]. The clustering algorithm operates by reading a sequence of documents, in our case a list of passages sorted by their scores in decreasing order, and making a decision for each document either to add it to one of the existing clusters (with or without updating the cluster's profile), or to start a new cluster. After clustering the top 1000 passages, we constructed the lists of passages to be used for query expansion by selecting first the passages in the cluster created as the first and continuing by adding passages from the clusters created later, until we reached the limit of 100 passages. Table 2 summarizes the results of the clustering experiments for both document- and passage- based second pass scoring. We used the clustering technique for our query description field based run only.

3 Spoken Document Retrieval Track

Our participation in the SDR track consisted of the reference (R1) and baseline (B1) runs. The text pre-processing and scoring techniques in our SDR experiments were based on those applied in our ad hoc entry and described in section 2. Bigram counts were collected for non-stop word pairs including pairs separated by a stop word. The number of top scoring documents used for query

	reference (R1)		baseline (B1)	
	AveP	P20	AveP	P20
pass1	0.4154	0.3940	0.3690	0.3660
pass2	0.4894	0.4530	0.4669	0.4470

Table 3: Spoken document retrieval results.

expansion was reduced to 60 to adjust for smaller size of the database.

We also tried applying a translation model to reduce the impact of speech recognition errors on the performance of the information retrieval system. In this view, there are two languages: the corpus of automatically transcribed data is considered to be one language of a parallel corpus, and a separately available corpus of manual transcriptions (of the same broadcast stories) is considered to be a separate language in a parallel corpus. Then retrieval of automatic transcriptions of broadcast news is considered to be a problem in cross-language information retrieval, since the queries (being free of speech recognition errors) more closely resemble the manual transcriptions. We then trained a statistical machine translation model of the type described in [9] to translate the *documents* from the language of automatically transcribed data into the language of hand-transcribed data. The test corpus was processed with this translation model, correcting some of the recognition errors and establishing cleaner text features to be used by the information retrieval system.

The training data was extracted from the January '98 part of the TDT2 corpus [7], which predates the SDR corpus. For the purpose of building the translation model, the output of the BBN speech recognizer served as the source language, close-captioning/manual transcripts being used as the target language. We aligned the source and target data sets at the level of sentences to form a parallel corpus. The translation model was trained on morphed representation of the corpus. We emphasize that manual transcriptions were used only in the training, not in the decoding phase.

Having trained the translation model, we applied it to translate the data produced by the BBN recognizer. Both the original and translated databases were indexed and scored separately with respect to the evaluation queries. We computed the final document scores as a linear combination of the scores of original and translated versions of the individual documents. Fig. 1 contains the average precision values for various relative weight combinations, showing a minor improvement achieved by incorporating a translation model in the system. The results of our SDR runs based on topics 74 to 123 are summarized in Table 3.

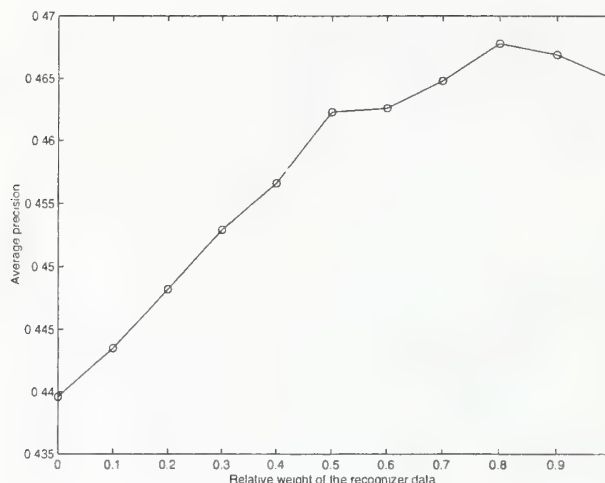


Figure 1: Combining the results based on ASR and translated data

4 Cross-language Track

4.1 Introduction

IBM's participation in the cross-language track at TREC-7 involved experiments with all four document languages : English, French, German, and Italian, and two of the query languages. Two experiments (*ibmcl8ea* and *ibmcl8ec*) were submitted based on the English queries, and two experiments (*ibmcl8fa* and *ibmcl8fc*) were based on the French queries. Our system is a composite system: we do initial retrievals for each language pair of interest, and then we merge the appropriate runs. The two experiments for each query language differed only in the merging strategy, not in the initial retrievals. The techniques studied here would also have been applicable to other query languages. All four runs used the long form of the queries. ("Long" queries used all three fields, <Title>, <Description>, and <Narrative>.) All query processing was fully automatic. We varied our strategy somewhat between the French and English query experiments. An important theme of our experiments has been that the widely varying availability and quality of bilingual resources (parallel and comparable corpora) requires that IR systems vary their strategy between language pairs accordingly. A second unifying theme of these experiments is the extensive use of statistical methods, reflecting the long history of statistical approaches to machine translation in our group. [6] In fact, all bilingual dictionaries and translation models used in these runs were learned automatically from corpora.

5 System Description

IBM's multilingual retrieval system is a composite system: a ranked list of potentially relevant documents is retrieved separately in each document language, and then these lists are merged. Because of the available bilingual resources, the retrieval engines associated with each language pair vary somewhat between language pairs. The *EqFd* is a hybrid query-translation document translation retrieval system, as described in [10], using the statistical machine translation algorithm described in [9]. Both the *English* \Rightarrow *French* query translation and the *French* \Rightarrow *English* document translation were trained from a parallel corpus (the Canadian Hansards.) The *FqEd* retrieval system is identical, except for the interchange of *French* and *English*. The *FqGd* system is also a hybrid query-translation document-translation IR systems, but the underlying *French* \Rightarrow *German* and *German* \Rightarrow *French* statistical translation models are trained from a comparable corpus (the SDA newswire itself), not a parallel corpus. The alignment of the comparable corpus was described in [3]. The *FqId* system is identical. The *EqGd* system is implemented using *French* as *pivot language*: we use the *EqFd* system to retrieve French documents, automatically constructed a French query from these documents, and then use the *FqGd* to retrieve German documents based on the artificial French query, as described in [3].

6 Results by Language Pairs

Because the IBM multilingual retrieval system is a composite system, it is important to observe individual aspects of our system's performance separately prior to merging. The most important aspects are the performance on the eight language pairs (systems for both French and English queries were submitted.) Results by query language and document language are shown in Tables 5. We also contrast the performance of the English query and French query systems on individual queries in the scatterplot in Fig. 2. Finally, we also contrast our systems performance on two subsets of the queries, which will have important consequences in the final merging. In analyzing the results of TREC-7, we noted that a significant fraction of the queries concern local European events, and these events are under-reported in the AP newswire. Furthermore, these queries can be automatically recognized, with reasonable reliability, by whether they specifically mention the name of a European country. This effect is shown in table 4 in which we denote the set of queries mentioning a European country *E* and the remainder of the queries *nE*. The same queries were identified as mentioning a European country in both the English-query and French-query experiments, although this need not have been the case if the human translations of the provided queries had been looser. We suspect that this effect also correlates with the country in which the query was originally constructed, but we

document language	$ E $	$ nE $	$ total $
English	140 (14.6%)	816	956
French	192 (33.2%)	386	578
German	327 (45.6%)	390	717
Italian	51 (30.0%)	119	170

Table 4: Number of relevant documents by document language and query subset

document language	AveP (Fr)	P20	AveP (Eng.)	P20
English	0.2952	0.3357	0.3049	0.3375
French	0.4706	0.3857	0.4186	0.3804
German	0.3142	0.3268	0.2559	0.2839
Italian	0.2788	0.1357	0.2221	0.1357

Table 5: Results by language pair

have not attempted to guess which queries were constructed in which countries.

7 Importance of Merging

Our merging strategy is to estimate the probability of relevance p of each document as a function $p(R, l_d, q)$ of the rank R that the document is retrieved by the systems for document language l_d , and also to allow this probability to depend upon features of the query q . The merging strategy as we formulate it here applies only to the merging of disjoint sets of documents. We have observed that the average precision at given rank R of information retrieval system is an approximately linear function of $\log(R)$ and we can use this linearity to form a two-parameter estimate of p for that system and set of queries [3]. We have a different estimate of p for each language pair. We also have a separate estimate for the query subsets E and nE (queries mentioning a European countries, and those that do not, respectively) and we find that this results in a slight improvement in performance over the single estimate for all queries. This strategy makes only the shallowest use of information about the query and the documents and it retrieves: other information, such as the IR engine's score of the document with respect to the query has not proven beneficial. Since the average precisions for this year's queries are significantly lower than last year's, we can test the sensitivity of the overall average precision to the parameterization of the merging strategy by tuning our merging strategy to this year's queries. We

query language	submission	merging	AveP
French	ibmcl8fa	$p(R, l_d, q)$	0.2613
French	ibmcl8fc	$p(R, l_d)$	0.2600
English	ibmcl8ea	$p(R, l_d, q)$	0.2559
English	ibmcl8ec	$p(R, l_d)$	0.2515

Table 6: Results by merging strategy

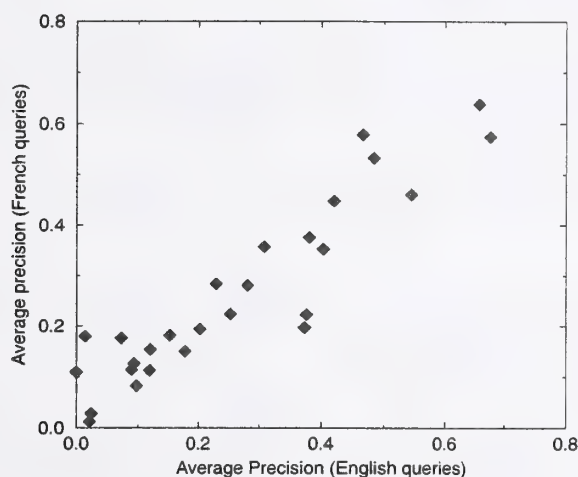


Figure 2: Scatterplot of average precision on English queries vs. French queries

find an approximate 10% improvement (average precision = 0.2803 on French queries.) These results are shown in Table 7.

8 Acknowledgments

This work is supported by NIST grant no. 70NANB5H1174.

References

- [1] S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, M. Gatford, "Okapi at TREC-3" in *Proceedings of the Third Text REtrieval Conference (TREC-3)* ed. by D.K. Harman. NIST Special Publication 500-225, 1995.

- [2] M. Franz and S. Roukos, "TREC-6 Ad-hoc Retrieval", in *Proceedings of the Sixth Text REtrieval Conference (TREC-6)* ed. by E. M. Vorhees and D.K. Harman. NIST Special Publication 500-240, 1998.
- [3] M. Franz, J.S. McCarley, S. Roukos, "Ad hoc and Multilingual Information Retrieval at IBM", in *Proceedings of the Seventh Text REtrieval Conference (TREC-7)* ed. by E. M. Vorhees and D.K. Harman. NIST Special Publication 500-242, 1999.
- [4] B. Merialdo, "Tagging text with a probabilistic model" in *Proceedings of the IBM Natural Language ITL*, Paris, France, pp. 161-172, 1990.
- [5] J. Xu and W. B. Croft, "Query Expansion Using Local and Global Document Analysis", in *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, pp. 4-11, 1996.
- [6] P. F. Brown et al. "The mathematics of statistical machine translation: Parameter estimation", *Computational Linguistics*, 19 (2), 263-311, June 1993.
- [7] S. Dharanipragada, M. Franz, J.S. McCarley, S. Roukos, T. Ward, "Story Segmentation and Topic Detection in The Broadcast News Domain", in *Proceedings of the DARPA Broadcast News Workshop*, 1999.
- [8] S. Dharanipragada, M. Franz, J.S. McCarley, S. Roukos, T. Ward, "Story Segmentation and Topic Detection for Recognized Speech", in *Proceedings of the Sixth European Conference on Speech Communication and Technology*, 1999.
- [9] J.S. McCarley and S. Roukos, "Fast Document Translation for Cross-Language Information Retrieval", in *Machine Translation and the Information Soup* ed. by D. Farwell, L. Gerber, and E. Hovy., 1998.
- [10] J.S. McCarley, "Should we Translate the Documents or the Queries in Cross-Language Information Retrieval?", in *37th Annual Meeting of the Association for Computational Linguistics* College Park, MD, 1999.

The Use of Predictive Annotation for Question Answering in TREC8

John Prager, Dragomir Radev*, Eric Brown, Anni Coden
IBM TJ Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
{jprager,ewb,anni}@us.ibm.com
*radev@umich.edu

Valerie Samn
Teachers College
Columbia University
New York, NY 10027
vs115@columbia.edu

ABSTRACT

This paper introduces the technique of Predictive Annotation, a methodology for indexing texts for retrieval aimed at answering fact-seeking questions. The essence of the approach can be stated simply: index the answers. This is done by establishing about 20 classes of objects that can be identified in text by shallow parsing, and by annotating and indexing the text with these labels, which we call QA-Tokens. Given a question, its class is identified and the question is modified accordingly to include the appropriate token(s). The search engine is modified to rank and return short passages of text rather than documents. The QA-Tokens are used in later stages of analysis to extract the supposed answers from these returned passages. Finally, all potential answers are ranked using a novel formula, which determines which ones among them are most likely to be correct.

1. INTRODUCTION

For question-answering, system designers have the choice of using technology from Information Retrieval or Natural Language Processing, or some combination thereof [1,3,4].

Information Retrieval systems employing traditional search engines are efficient but suffer from the fact that they generally return documents rather than answer passages, let alone precise answers, and that the documents that are returned are ranked based on frequency of occurrence of query terms rather than any correspondence with what the query is seeking. Natural Language Processing systems can to a greater or lesser extent overcome the problem of semantic matching, but are inherently expensive; this can make processing a database the size of that in the TREC8 exercise inherently intractable.

Our approach attempts a middle ground. Our group's principal experience has been with traditional IR systems,

but also with building text-analysis systems such as TEXTRACT [6,7].

We decided to build a modified search engine that works in conjunction with shallow NLP of the text. We call our technology Predictive Annotation since we identify and annotate in the text generalizations of the base terms; these annotations are designed to correspond to the terminology used in questions.

Our approach is based on the following five observations of questions seeking facts (as opposed to How and Why questions that seek procedural answers) and the texts that typically contain them.

(1) In documents that contain the answers, the query terms that occur there tend to occur in close proximity to each other. They will typically occur within passages of 2-3 sentences - often within one sentence. It is only the single occurrence of a query term in this passage that seems to count; other occurrences elsewhere are more-or-less irrelevant.

(2) The answers to fact-seeking questions are usually phrases: noun phrases ("President Clinton"), prepositional phrases ("in the mountains") and adverbial phrases ("today").

(3) These phrases can be typed by a set of a dozen or so labels (such as PERSON\$, PLACE\$, MONEY\$, LENGTH\$,...).

(4) These categories correspond to question words ("Who", "Where", "How much", "How long", ...).

(5) The phrases can be identified in text by simple pattern-matching techniques.

In this paper, we describe the different stages that our system goes through to select answers and we present our results on the official TREC evaluation [1].

2. SYSTEM OPERATION OVERVIEW

It will be clear by now that phrases play a prominent role in our system. It is no coincidence that, for the most part, phrases can be detected in text with a relatively simple pattern-matching algorithm, certainly a requirement that falls far short of full natural-language understanding.

The implementation of the solution is expressed in modifications to three of the components of a traditional search engine solution, namely query-analysis, text parsing and indexing, and scoring.

- (1) The user queries are pre-processed with question-words replaced by an invented set of labels we call QA-Tokens.
- (2) The text to be indexed is analysed for phrases of certain types. The QA-Tokens corresponding to these types are indexed in addition to the base terms.
- (3) The matching process scores short sequences of sentences rather than documents, with weighting much coarser than the traditional $tf \cdot idf$ or its variants.

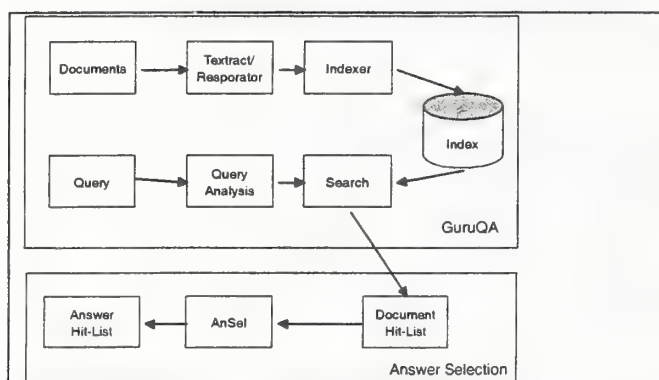


Figure 1: System Architecture

3. QUERY PROCESSING

The query analysis is enhanced by developing a set of question-templates that are matched against the user's query, with substitution of certain query terms with our special QA-Tokens that correspond to the phrase labels mentioned above. So for example, the pattern "where..." causes the word "where" to be replaced with **PLACES\$**. The pattern "how much does ... cost" causes those terms to be replaced with **MONEY\$**. The pattern "how old ..." causes a replacement with **AGE\$**. The base set of such labels is: **PLACES\$, PERSON\$, ROLE\$, NAME\$, ORGANIZATION\$, DURATION\$, AGE\$, DATE\$, TIME\$, VOLUME\$, AREA\$, LENGTH\$, WEIGHT\$, NUMBER\$, METHOD\$, MOST\$, RATE\$** and **MONEY\$**. More specific versions of these, such as **STATES\$, COUNTRY\$, CITY\$, YEAR\$** can be used as long as the

phrase analyser (discussed below) can recognize such quantities. The use of QA-Tokens to represent potential matching quantities in text is similar to, but much more general than the noun-phrase matching of Kupiec[5].

In some patterns, the entire set of matching terms is removed, as when "How much does <any text> cost" gets transformed to "**MONEY\$** <any text>". In some other patterns, though, one or more of the matching terms is retained, as when "What is the population of <any text>" gets transformed to "**NUMBER\$** population <any text>". The set of these patterns currently numbers around 180. The QA-Token set currently numbers around 20. The QA-Tokens are listed in Figure 2, along with the question-words they can correspond to (this mapping is close to, but not exactly, one-to-one) and sample patterns in text they are aiming to discover.

QA-Token	Question type	Example
PLACES\$	Where	In the Rocky Mountains
COUNTRY\$	Where/What country	United Kingdom
STATES\$	Where/What state	Massachusetts
PERSON\$	Who	Albert Einstein
ROLE\$	Who	Doctor
NAME\$	Who/What/Which	The Shakespeare Festival
ORG\$	Who/What	The US Post Office
DURATION\$	How long	For 5 centuries
AGE\$	How old	30 years old
YEAR\$	When/What year	1999
TIME\$	When	In the afternoon
DATE\$	When/What date	July 4 th , 1776
VOLUME\$	How big	3 gallons
AREA\$	How big	4 square inches
LENGTH\$	How big/long/high	3 miles
WEIGHT\$	How big/heavy	25 tons
NUMBER\$	How many	1,234.5
METHOD\$	How	By rubbing
RATE\$	How much	50 per cent
MONEY\$	How much	4 million dollars

Figure 2: QA-tokens

A synonym operator **@SYN()** is used to deal with cases where a question could be validly matched against more than one type of phrase. Thus a "who" question could

match a proper name, a profession or an organization, so will generate @SYN(PERSON\$, ROLE\$, ORG\$, NAME\$) in the modified query.

The document collection is analyzed by TEXTTRACT[6,7] prior to indexing; one of the outputs of this process is a dictionary containing the collection vocabulary. This is used in query processing to discover proper names in the query and optionally to convert names and terms to their canonical forms. A subsystem of TEXTTRACT is used to convert common words to their lemma forms, and identify stop-words for removal (which happens AFTER the pattern-matching described above).

We do not weight terms in the index in the traditional IR fashion; instead we weight selected query terms, and specify this in the query syntax by means of a weight operator @WEIGHT(). We use a very coarse granularity of weighting. We choose a base weight of 100 for common words.

Proper names and other multi-word terms in the query are rarer than individual words and so their presence in answer sentences gives more confidence that the sentence is correct than the presence of single terms from the query, all other things being equal, so should be weighted higher. We use a weight of 200 for such items.

Now, any proposed answer text is no answer if it doesn't contain a type-compatible match to a special query-token in the query, so special query tokens should be weighted higher than any other words in the query. We use a weight of 400 for the QA-Tokens.

Some of the alternatives in the @SYN-groups may be more desirable than others. For example, "when" might generate @SYN(DATE\$, TIME\$), where DATE\$ matches specific dates (e.g. "July 4th, 1776") but TIME\$ matches more general expressions ("in the afternoon"); a DATE\$ match is therefore usually more desirable than a TIME\$ match so might be weighted more. We currently don't differentially weight within a @SYN-group. However, we do order the elements in a @SYN-group in decreasing order of desirability; this order is considered in the final answer-selection phase when the passages returned from our search engine contain multiple contenders for "the answer".

We take into account the density of the matching words in a scored passage. Intuitively, since the query words all occur together in a short sentence (the query), so the closer together they occur in a text passage, all other things being equal, the more likely the text reflects the semantics of the query. Hence we calculate a density component to the passage's score in the range 0 to 99 (the latter representing the case of all matching terms being adjacent). This is added to weighted score of appearing query terms.

Finally, the query can be augmented by the @WIN operator through which we specify the target window size and whether matching in this window is to be *exclusive* or not. The window size is an integer representing the size in sentences of a moving window of text within which matching is attempted. We extended this approach to that of using a dynamic window, which uses the stated window size as an upper bound, but prefers sub-windows if they happen to contain the same matches as the larger window. This approach seems to overcome the need to apply iterative narrowing and broadening operations to the query as done by Kupiec[5].

The issue of exclusivity represents the desire to avoid having a QA-Token match a word in the text that is already matched with another query term. Thus if the query is "Whom did President Clinton meet" and the text states that "President Clinton met Tony Blair ...", we don't want the PERSON\$ token in the query to match with the PERSON\$ attached to President Clinton but rather the one attached to Tony Blair.

4. INDEXING

Our extensions to indexing are somewhat similar to the Predictive Indexing of [4], which adds to the index related terms discovered via WordNet[9]. Both techniques are aimed at increasing recall. Their approach does this by adding related terms to the index, 'in case' the user phrases questions with that particular. Instead, we annotate with the QA-Token that stands for the conceptual category of the index term, and is generated by our query-analysis process.

The indexer runs its own pattern template matcher against the text in the documents to be indexed. For each of the phrase types, a set of patterns needs to be developed. For example, the following are some of the TIME\$ phrases:

in the afternoon
in the morning
in :CARDINAL hours

(where :CARDINAL is a cardinal number), and so on. Clearly to avoid a huge list (consider that instead of "hours" in the last example, almost any word indicating a period of time could be substituted), a mechanism for concisely expressing such variants and for efficient performance of the matching is desirable. We built such a system, which we call Resporator, as another annotator for TEXTTRACT.

Whenever the indexer succeeds in matching a phrase pattern template in text, the corresponding QA-Token (such as TIME\$ or PLACE\$) is generated and indexed at that point in the document, along with the individual terms that comprised the phrase. We call this process of adding ex-

tra indexing terms annotation. All original terms in the document are indexed in the usual way too.

5. SEARCHING

The search engine operates essentially by the usual bag-of-words matching technique, but is subtly affected by the presence of the QA-Tokens. Thus the query: "When did the Challenger explode" gets translated on query analysis to the bag { @SYN(DATE\$, TIME\$) Challenger explode } which matches best against locations in the index that contain (exactly or variants of) the word Challenger, the word explode and either a DATE\$ or a TIME\$ token, meaning some phrasal expression of a time or date.

The QA-Tokens are matched after the other query terms in order to be able to enforce exclusivity. It is not required, though, that there be any QA-Token in the query at all; this is the situation which occurs when the query doesn't match any of our Query Patterns. The search engine operates in such cases just as it would if there were QA-Tokens, except that exclusivity is not an issue.

The final mentioned improvement is to the scoring algorithm. Search engines usually score documents based on how many of the query terms they contain and how often, combining contribution weights computed for each term based on document and collection-based statistics. It is our observation that when a document successfully answers a question, all of the components of the question are to be found together, usually within a passage of a sentence or two. The number of other occurrences of query terms elsewhere in the document does not seem to be a useful indicator of the passage's worth.

Thus we modify the scoring algorithm to score sentences (or short sequences of them) rather than documents. Due to the more severe filtering constraints imposed by this (i.e. that all, or most, query terms must occur in such a passage, rather than the document as a whole), then a less complicated scoring function, namely the weighting scheme described earlier, has been found to suffice.

6. ANSWER SELECTION

The earlier sections of this paper described how we retrieve relevant passages that may contain the correct answer to a query. The top 10 passages returned by the search engine at this stage consists of a large number (often more than 30 or 40) potential answers. The following three sections describe how we determine which ones among these answers are more likely.

6.1 Answer ranking

The TREC8 QA-Track requires participants to return sequences of text, which we call here spans, of length either 50 or 250 bytes. This part of the paper describes two systems, AnSel and Werlect, which are used independently of each other to extract these spans from the passages returned by GuruQA, and rank them. AnSel and Werlect use different approaches, which we describe, evaluate and compare and contrast. The output of either system consists of five text extracts per question that contain the likeliest answers to the questions.

GuruQA is first used to find the passages that are considered to be most relevant to the question and labels them with QA-tokens as shown below.

6.2 Sample Input to AnSel/Werlect

The role of answer selection is to decide which among the spans extracted by GuruQA are most likely to contain the precise answer to the questions. Figure 3 contains an example of the data structure passed from GuruQA to our answer selection modules. The example is taken from the official TREC evaluation questions.

```
<p><NUMBER>1</NUMBER></p>
<p><QUERY>Who is the author of the book, "The
Iron Lady: A Biography of Margaret
Thatcher"?</QUERY></p>
<p><PROCESSED_QUERY>@excwin(*dynamic*
@weight(200 *Iron_Lady) @weight(200 Biogra-
phy_of_Margaret_Thatcher) @weight(200 Marga-
ret) @weight(100 author) @weight(100 book)
@weight(100 iron) @weight(100 lady)
@weight(100 :) @weight(100 biography)
@weight(100 thatcher) @weight(400 @syn(PERSON$
NAME$)) )</PROCESSED_QUERY></p>
<p><DOC>LA090290-0118</DOC></p>
<p><SCORE>1020.8114</SCORE></p>
<TEXT><p>THE IRON LADY; A <span
class="NAME">Biography of Margaret
Thatcher</span> by <span class="PERSON">Hugo
Young</span> (<span class="ORG">Farrar ,
Straus & Giroux</span>) The central riddle
revealed here is why, as a woman in a man's
world, <span class="PERSON">Margaret
Thatcher</span> evinces such an exclusionary
attitude toward women.</p></TEXT>
```

Figure 3: Input sent from GuruQA to AnSel

The input consists of four items:

- a query (e.g., "Who is the author of the book "The Iron Lady: A Biography of Margaret Thatcher"?),
- a list of passages (one is shown above; it is surrounded with <TEXT> and </TEXT>),

- a list of annotated text spans within the passages, annotated with span types (QA-tokens), and
- the list of potential span types (or SYN-group) for the type of question recognized by Resporator (e.g., "PERSON\$ NAME\$" in the example above).

In Figure 3, we only showed the first passage retrieved by GuruQA. It contains five spans, of which three ("Biography of Margaret Thatcher", "Hugo Young", and "Margaret Thatcher") are of types included in the SYN-group for the question (PERSON NAME). The total output of GuruQA for this question includes five passages and a total of 14 potential spans (5 PERSONs and 9 NAMEs).

6.3 Sample Output of AnSel/Werlect

Our system has two outputs: one internal to the system and one that is submitted for evaluation.

6.3.1 Internal output

The internal output is a ranked list of spans as shown in Table 1. It represents a ranked list of the spans (potential answers) sent by GuruQA.

Score	Span
5.06	Hugo Young
-8.14	Biography of Margaret Thatcher
-13.60	David Williams
-18.00	Williams
-19.38	Sir Ronald Millar
-25.80	PP
-26.06	Santiago
-31.75	Oxford
-32.38	Maggie
-36.78	Seriously Rich
-42.68	FT
-198.34	Margaret Thatcher
-217.80	Thatcher
-234.55	Iron Lady

Table 1: Ranked potential answers to Question 1

While only the external output (see below) was required for the TREC evaluation, our system's internal output can be used in a variety of related applications. For example, we can highlight the actual span that we believe is the answer to the question within the context of the passage in which it appears. We can also perform frequency analyses based on the extracted spans.

6.3.2 External output

The external output is a ranked list of 50-byte and 250-byte extracts. These extracts are selected in a way to cover

the highest-ranked spans in the list of potential answers. Examples are given later in the paper.

7. ANALYSIS OF CORPUS AND QUESTION SETS

To train our system, we used the set of 38 training questions provided by NIST. In the rest of this paper, we will refer to these questions as TR38. The results presented in the evaluation section of this paper are based on the 200 test questions (also provided by NIST) which we were not allowed to look at until the official submission of our results. In this section we describe the corpora used for training and evaluation as well as the questions contained in the training and evaluation question sets.

7.1 Corpus analysis

The corpus used for both training and evaluation (see Table 2) consisted of approximately 2 GB of news articles from four equally represented sources: the Foreign Broadcast Information Service (FBIS), the Los Angeles Times (LA), the Financial Times (FT), and the Federal Register (FR). This corpus has been used as a standard in several past TREC text retrieval conferences.

	Year	Size in MB	No. of Docs
FBIS	1996	493	130,471
LA	1989-90	498	131,896
FT	1991-94	592	210,158
FR	1994	414	55,630

Table 2: Description of the corpus used

7.2 Training set TR38

The training set contained questions for which the answers were provided to us for system training and parameter estimation. Some sample questions are shown in Figure 4:

Question/Answer (T38)
Q: Who was Johnny Mathis' high school track coach? A: Lou Vasquez
Q: What year was the Magna Carta signed? A: 1215
Q: What two companies produce bovine somatotropin? A: Monsanto and Eli Lilly
Q: When did Nelson Mandela become president of South Africa? A: 10 May 1994

Figure 4: Sample questions and answers from TR38

7.3 Test set T200

The majority of the questions (see Figure 5) in T200 were not substantially different in style from these in TR38. The introduction of “why” and “how” questions as well as the wording of questions in the format “Name X” caused us some trouble because at the time we had no matching question templates. Other problems were caused by questions that require specific semantic types of answers, such as “star”, “designer”, “film”, and “export” for which our system didn’t contain extraction and labeling patterns¹. Other problems were because of oversights in our question templates and were easily fixed; for example, we were missing a pattern that matched “How rich ...” and generated the MONEY\$ token.

Question/Answer (T200)
Q: Who was chosen to be the first black chairman of the military Joint Chiefs of Staff? A: Colin Powell
Q: How tall is the Matterhorn? A: The institute revised the Matterhorn 's height to 14,776 feet 9 inches
Q: How tall is the replica of the Matterhorn at Disneyland? A: In fact he has climbed the 147-foot Matterhorn at Disneyland every week end for the last 3 1/2 years

Figure 5: Sample questions and answers from T200

Some examples of problematic questions are shown in Figure 6.

Q: Why did David Koresh ask the FBI for a word processor?
Q: Name the designer of the shoe that spawned millions of plastic imitations, known as "jellies".
Q: What are the Valdez Principles?
Q: Name a film that has won the Golden Bear in the Berlin Film Festival?
Q: What did Shostakovich write for Rostropovich?
Q: What is the term for the sum of all genetic material in a given organism?
Q: What is considered the costliest disaster the insurance industry has ever faced?
Q: What is Head Start?
Q: What was Agent Orange used for during the Vietnam War?
Q: What did John Hinckley do to impress Jodie Foster?

¹ Note that after the evaluation was officially over, we created new patterns in our system which cover some of these cases.

Q: What was the first Gilbert and Sullivan opera?
Q: What did Richard Feynman say upon hearing he would receive the Nobel Prize in Physics?
Q: How did Socrates die?
Q: Why are electric cars less efficient in the north-east than in California?

Figure 6: Some harder questions in T200

We performed an analysis of the most frequent types of questions that occur in the evaluation corpus. Table 3 contrasts the performance of our system’s best run on the different types of questions (represented as SYN-groups).

SYN-group	N	Score	Score/ N
PERSON NAME	30	16.5	55.0%
PLACE COUNTRY STATE NAME PLACEDDEF	21	7.08	33.7%
NAME	18	3.67	20.4%
DATE YEAR	18	5.31	29.5%
PERSON ORG NAME ROLE	19	4.62	24.3%
undefined	19	11.45	60.3%
NUMBER	18	8.00	44.4%
PLACE NAME PLACEDDEF	14	10.00	71.4%
PERSON ORG PLACE NAME PLACEDDEF	10	3.03	30.3%
MONEY RATE	6	1.50	25%
ORG NAME	4	1.25	31.2%
SIZE1	4	2.50	62.5%
SIZE1 DURATION	3	0.83	27.7%
STATE	3	2.00	66.7%
COUNTRY	3	1.33	44.3%
YEAR	2	1.00	50.0%
RATE	2	1.50	75.0%
TIME DURATION	1	0.00	0.0%
SIZE1 SIZE2	1	0.00	0.0%
DURATION TIME	1	0.33	33.3%
DATE	1	0	0.00%

Table 3: Performance of A250 on different types of SYN-groups

8. RANKING SPANS

The two answer ranking systems, AnSel and Werlect use different algorithms which we describe in this section.

8.1 AnSel

The first algorithm that we used is called AnSel (ANSwer SElection). It is essentially an optimization algorithm that uses 7 predictive variables to describe how likely a given span is to be the correct answer to a given question. The

predictive variables are illustrated with examples related to the sample question number 10001 from TR38 “Who

was Johnny Mathis’ high school track coach?” and the top potential answers to which are shown in Table 4.

Span	Type	Number	Rspanno	Count	Noting	Type	Avgdst	Sscore	TOTAL
Ollie Matson	PERSON	3	3	6	2	1	12	0.02507	-7.53
<i>Lou Vasquez</i>	<i>PERSON</i>	<i>1</i>	<i>1</i>	<i>6</i>	<i>2</i>	<i>1</i>	<i>16</i>	<i>0.02507</i>	<i>-9.93</i>
Tim O'Donohue	PERSON	17	1	4	2	1	8	0.02257	-12.57
Athletic Director Dave Cowen	PERSON	23	6	4	4	1	11	0.02257	-15.87
Johnny Ceballos	PERSON	22	5	4	1	1	9	0.02257	-19.07
Civic Center Director Martin Durham	PERSON	13	1	2	5	1	16	0.02505	-19.36
Johnny Hodges	PERSON	25	2	4	1	1	15	0.02256	-25.22
Derric Evans	PERSON	33	4	4	2	1	14	0.02256	-25.37
NEWSWIRE Johnny Majors	PERSON	30	1	4	2	1	17	0.02256	-25.47
Woodbridge High School	ORG	18	2	4	1	2	6	0.02257	-28.37
Evan	PERSON	37	6	4	1	1	14	0.02256	-29.57
Gary Edwards	PERSON	38	7	4	2	1	17	0.02256	-30.87
O.J. Simpson	NAME	2	2	6	2	3	12	0.02507	-37.40

Table 4: Feature set and span rankings for a sample question

8.1.1 Feature selection

The seven span features described below were found to correlate with the YES/NO categories of the training data. As an illustration, we use the answer to training question number 10001.

Number: position of the span among all spans returned. Example: “*Lou Vasquez*” was the first span returned by GuruQA on the sample question.

Rspanno: position of the span among all spans returned within the current passage.

Count: number of spans of any span class retrieved within the current passage.

Noting: the number of words in the span that do not appear in the query. Example: **Noting** (“*Woodbridge high school*”) = 1, because both “high” and “school” appear in the query while “Woodbridge” does not. It is set to -100 when the actual value is 0.

Type: the position of the span type in the list of potential span types. Example: **Type** (“*Lou Vasquez*”) = 1, because the span type of “*Lou Vasquez*”, namely “PERSON” appears first in the SYN-group, “PERSON ORG NAME ROLE”.

Avgdst: the average distance in words between the beginning of the span and the words in the query that also appear in the passage. Example: given the passage “*Tim O'Donohue, Woodbridge High School's varsity baseball coach, resigned Monday and will be replaced by assistant Johnny Ceballos, Athletic Director Dave Cowen said.*” and the span “*Tim O'Donohue*”, the value of **avgdst** is equal to 8.

Sscore: passage relevance as computed by GuruQA.

8.1.2 AnSel Training Algorithm

The TOTAL score for a given potential answer is computed as a linear combination of the features described in the previous subsection:

$$\text{TOTAL} = \sum w_i f_i$$

The algorithm used by the training component of AnSel learn the weights used in the formula is shown in Figure 7.

For each <question,span> tuple in training set:

1. Compute features for each span
2. Compute TOTAL score for each span using current set of weights

Repeat

3. Compute performance on training set
4. Adjust weights w_i

Until performance > threshold

5. Store weights for use in ranking

Figure 7: Algorithm used by AnSel to learn the weights

8.1.3 AnSel Ranking Algorithm

Once AnSel has learned the weights during the training stage, it can be used to rank potential answers to other questions. The algorithm used is shown in Figure 8.

For each question in test set:

1. Compute features for each span
2. Compute TOTAL score for each span using weights w_i
3. Rank spans
4. Let `current_span` = highest ranked span, and let `answer_set` = {}

Repeat

5. Let `current_extract` = extract of 50 (or 250) bytes centered around `current_span`
6. Skip current extract if already included in `answer_set`
7. Insert `current_extract` in `answer_set`

Until `answer_set` contains five extracts

8. Output `answer_set`

Figure 8: Algorithm used by AnSel for ranking potential answers

8.1.4 Example system run

For the question "Who was Johnny Mathis' high school track coach?", GuruQA retrieved a total of 23 spans (12 were tagged by Resporator as "PERSON", seven as "NAME", three as "ROLE", and one as "ORG"). The signature of the question indicates that these are the four possible span types for the answer, ordered PERSON, ORG, NAME, ROLE from the likeliest to the least likely.

The computed scores are based on the following scoring formula:

$$\text{TOTAL}(\text{span}) = -0.3 * \text{number} - 0.5 * \text{rspanno} + 3.0 * \text{count} + 2.0 * \text{notinq} - 15.0 * \text{types} - 1.0 * \text{avgdst} + 1.5 * \text{sscore}$$

After AnSel has ranked all potential answers, it extracts a set of 50- and 250-byte passages that cover the top answers in the list. The actual extracts are shown in Figure 9 and Figure 10.

Document ID	Score	Extract
LA053189-0069	892.5	of O.J. Simpson , Ollie Matson and Johnny Mathis
LA053189-0069	890.1	Lou Vasquez , track coach of O.J. Simpson , Ollie
LA060889-0181	887.4	Tim O'Donohue , Woodbridge High School 's varsity
LA060889-0181	884.1	nny Ceballos , Athletic Director Dave Cowen said.
LA060889-0181	880.9	aced by assistant Johnny Ceballos , Athletic Direc

Figure 9: Fifty-byte extracts

Document ID	Score	Extract
LA053189-0069	892.5	Lou Vasquez , track coach of O.J. Simpson , Ollie Matson and Johnny Mathis during his 32-year career, died Saturday while at his South Lake Tahoe vacation cabin, it was announced Tuesday . He was 68 . His Washington High school teams won five consecu
LA060889-0181	887.4	Tim O'Donohue , Woodbridge High School 's varsity baseball coach , resigned Monday and will be replaced by assistant Johnny Ceballos , Athletic Director Dave Cowen said.
LA062090-0017	880.6	Civic Center Director Martin Durham said Mathis was to have entered the parking lot in a convertible Rolls-Royce to cut the ribbon for the dedication of Johnny Mathis Boulevard .
LA052390-0122	874.8	Ellington liked what he heard. Johnny Hodges was quitting the band and Morgan was invited to replace him, but he couldn't leave high school to go on the road. The new album's title track and " In a Sentimental Mood " are Morgan 's most recent homages
LA062389-0083	874.6	NEWSWIRE Johnny Majors , Tennessee football coach , said that prize recruit Derric Evans will not be allowed to play for the Volunteers because of his arrest Tuesday night in Dallas . Evans and a high school teammate, Gary Edwards , were charged with

Figure 10: Two-hundred-and-fifty-byte extracts

8.2 Werlect

The algorithm used to create VS50 and VS250 made use of many of the same features of noun phrases (spans), that were used in AnSel, but employed a different ranking scheme. We refer to this sister algorithm as Werlect (anSWER seLECTION).

8.2.1 Approach

Unlike AnSel, the algorithm developed for Werlect was based not on a simple linear function, but a two-step, rule-based process approximating a function that included interaction between variables. In the first stage of this algorithm, we assign a rank to every relevant noun phrase within each sentence according to how likely it is to be the target answer. Next, we generated and ranked each N-byte ($N = 50$ or 250) fragment, based on the sentence score given by Guru-QA, measures of the fragment's relevance, and the ranks of its component noun phrases. Werlect also differed from AnSel in its development in that there was no training algorithm, but was instead developed through manual trial-and-error, optimizing for the TR38 questions.

8.2.2 Step One: Feature selection

The first task of Werlect's two-step analysis is to rank each noun phrase, or span, for each hit returned by Guru-QA. In addition to the noun phrase's type, three main features were used to rank the noun phrases in each sentence. It is hypothesized that the target answer 1) is more likely to appear in multiple hits; 2) may contain, in some part, some of the query terms, and 3) is likely to be closer in proximity to matching query terms. With these rules, we hoped to identify the best noun phrase among several selected within one hit, and when necessary, promote a span higher than the rank awarded by Guru-QA.

Thus, the noun phrase features considered in Werlect are analogous to those used in AnSel, including **Type** (the position of the span type in the list of potential span types), **Avgdst** (the average distance in words between the beginning of the span and the words in the query that also appear in the passage), **Sscore** (passage relevance as computed by Textract and Resporator). Two additional features were also taken into account:

NotinqW: a modified version of **Notinq** (the number of words in the span that do not appear in the query). As in AnSel, spans that are completely contained in the query are given a rank of 0. However, partial matches are weighted favorably.

Frequency: how often the span occurs across different passages (not to be confused with AnSel's **Count**, which refers to the number of occurrences only within the current passage)

Examples of the isolated effects of Avgdst, NotinqW, and Frequency on answer selection are presented below.

Proximity (AveDist)

We hypothesize that the target answer is closer in proximity to the matched terms. Figure 11 shows a candidate answer that contains four noun phrases of the desired type, NUMBER. The noun phrases appear in bold, with their respective reciprocal average distances from the matched terms. The correct answer, 60 million, has the highest reciprocal average distance (.437) of the four noun phrases.

"What is the number of buffaloes thought to have been living in North America when Columbus landed in 1492?"

". . . there are between **60,000** (.318) to **80,000** (.336) head of bison in America. . . . That's not many compared to the estimated **60 million** (.437) that inhabited North America when Columbus discovered it in 1492, or even compared to the **20 million** (.25) that still roamed the Great plains in the 1850's."

Figure 11: Question and Text passage with four potential answers

This criterion effectively helps to identify one potential answer over the others within a single passage containing several candidates.

Relevance (NotinqW)

Although we know that a noun phrase that is completely contained in the query cannot be the answer, a noun phrase that contains part of the query may be more relevant. For example, if the question asks, "Who was Lincoln's Secretary of State?" a noun phrase that contains "Secretary of State" is more likely to be the answer than one that does not. In this example, the noun phrase, "Secretary of State William Seward" is the most likely candidate, based on this criterion.

This criterion also seems to play in a role in the event that Resporator fails to identify relevant phrase types. For example, in the training question, "What shape is a porpoise's tooth?" the phrase "spade-shaped" is chosen from among all nouns and adjectives of the sentences returned by Guru-QA.

Frequency

We hypothesize that a correct answer is more likely to occur in multiple hits than incorrect answers. For example, the test question, "How many lives were lost in the Pan Am crash in Lockerbie, Scotland?" resulted in four answers in the first two sentences returned by Guru-QA. Table 5: Influence of frequency on final 50-byte span rank shows the frequencies of each term, and their eventual influence on the span rank.

Initial Sentence Rank	Noun Phrase	Frequency	Span Rank
1	Two	5	2
1	365 million	1	3
1	11	1	4
2	270	7	1

Table 5: Influence of frequency on final 50-byte span rank

Without considering the criterion described here, the competing noun phrases from a single answer, such as "two," "365 million," and "11," are tied, and are essentially arbitrarily ranked in first, second, and third place. Taking the frequency into consideration, the phrase "two" is awarded the top rank within that answer, yielding a span rank of 2 out of all possible 50-byte spans. However, the correct answer, "270," occurs seven times among the top ten answers returned by the search engine, serving to promote the fragment that spans it to first place.

8.2.3 Step two: Ranking the Sentence Spans

After each relevant noun phrase is assigned a rank, spans of 50 (or 250) bytes of all answers are created. Then, to each is assigned a score that is equal to the sum of the noun phrase ranks plus additional points for other words that match the query. A fuller account of this algorithm will be reported elsewhere.

9. EVALUATION

In this section, we describe the performance of our system on the training data and on the test data. For the test data, we refer to the four runs that we submitted officially. These four runs are labeled as follows: A50 (AnSel on 50 bytes), A250 (AnSel on 250 bytes), W50 (Werlect on 50 bytes), and W250 (Werlect on 250 bytes).

9.1 Evaluation scheme

For each question, the performance score is computed as the reciprocal answer rank (RAR) of the first correct answer given by the. To compute the overall performance of the system, we use the Mean Reciprocal Answer Rank (MRAR):

$$\text{MRAR} = 1/n (\sum_i 1/\text{rank}_i)$$

9.2 Performance on TR38

The system performance on TR38 is shown in Table 6. We were able to get the correct answer in first place for 14

questions out of 38 while answering 7 others within the next four places.

	First	Second	Third	Fourth	Fifth	TOTAL (MRAR)
# cases	14	2	2	1	2	21
Points	14.00	1.00	0.67	0.25	0.40	16.32 (.77)

Table 6: Performance on TR38

9.3 Performance on official evaluation data

The performance of A50 and 250 on T200 are shown in Table 7 and Table 8, with MRAR of .32 and .43 respectively.

	First	Second	Third	Fourth	Fifth	TOTAL (MRAR)
# cases	49	15	11	9	4	198
Points	49.00	7.50	3.67	2.25	0.80	63.22 (.32)

Table 7: Performance of A50 on T200

	First	Second	Third	Fourth	Fifth	TOTAL (MRAR)
# cases	71	16	11	6	5	198
Points	71.00	8.00	3.67	1.50	1.00	85.17 (.43)

Table 8: Performance of A250 on T200

The next table shows some statistics on our two best runs (for 50 and 250 bytes, respectively).

While on the whole Ansel performed better than Werlect, we discovered that the relative performances reversed on those questions where our question analysis was unable to identify a suitable QA_Token. This phenomenon suggests a hybrid system should be investigated.

To give a better idea of the performance of our system, we split the 198 questions into 20 groups of 10 questions (or 9 questions in the two cases in which the evaluators removed questions from the original 200-question set). Our performance on a group of questions ranged from 0.87 to 5.50 points for the 50-byte run (A50) and from 1.98 to 7.5 points for the 250-byte run (A250), as shown in Table 9.

	50 bytes	250 bytes
n	20	20
Avg	3.19	4.30
Min	0.87	1.98
Max	5.50	7.50
Std Dev	1.17	1.27

Table 9: Performance on groups of ten questions

The final evaluation (included in Table 10) shows how well our system did compared to the rest of the 25 participants in the TREC Q&A evaluation.

Run	Median Average	Our Average	# Times Our Run > Median	# Times Our Run = Median	# Times Our Run < Median
W50	0.12	0.28	56	126	16
A50	0.12	0.32	72	112	14
W250	0.29	0.39	60	106	32
A250	0.29	0.43	66	110	22

Table 10: Comparison of our entries and the other participants

10. CONCLUSION

We presented a new technique for finding answers to natural language questions using text corpora as reference.

We showed that a span-centered approach to question answering can deliver very good results.

We described seven features that correlate with the plausibility of a given text span being a good answer to a question. We showed that a linear combination of these features performs well on the task of ranking text spans by the estimated relevance to the natural language question.

In the future, we plan to concentrate on getting better categories of text spans in order to provide fine-grained matches between question types and span types. We also intend to perform large-scale parameter learning.

We know we need to expand the set of question templates for existing QA-Tokens, as well as add more QA-Tokens and corresponding templates for a broader set of syntactic quantities.

Finally, we plan to investigate how language reuse and regeneration (LRR) [2] techniques can be used to provide contextual answers to natural language questions in a dialogue environment.

11. REFERENCES

- [1] TREC Q&A Evaluation official Web site: <http://www.research.att.com/~singhal/qa-track.html>
- [2] Dragomir R. Radev. "Language Reuse and Regeneration: Generating Natural Language Summaries from Multiple On-Line Sources", PhD thesis, Department of Computer Science, Columbia University, New York, October 1998.
- [3] AAAI Fall Symposium on Question Answering, North Falmouth, MA, 1999.
- [4] V.A. Kulyukin, K.J. Hammond, and R.D. Burke. "Answering Questions for an Organization Online", Proceedings of AAAI'98.
- [5] Julian Kupiec "Murax: A Robust Linguistic Approach for Question Answering Using an On-line Encyclopaedia", Proceedings of SIGIR'93.
- [6] Roy Byrd and Yael Ravin. "Identifying and Extracting Relations in Text", Proceedings of NLDB 99, Klagenfurt, Austria.
- [7] Nina Wacholder and Yael Ravin and Misook Choi. "Disambiguation of Proper Names in Text", Proceedings of ANLP'97. Washington, DC, April 1997.
- [8] Dragomir Radev and Kathleen McKeown. "Building a generation knowledge source using internet-accessible newswire". Proceedings of ANLP'97. Washington, DC, April 1997.
- [9] George Miller. "WordNet: A Lexical Database for English", Communications of the ACM 38(11) pp 39-41, 1995.

IIT at TREC-8: Improving Baseline Precision

M. Catherine McCabe
Advanced Analytic Tools
Washington, DC
catherm@ir.iit.edu

David O. Holmes
NCR Corporation
Rockville, MD
david.holmes@washingtondc.ncr.com

Kenneth L. Alford
US Army
Springfield, VA
ken4sher@erols.com

Abdur Chowdhury
IIT Research Institute
Rockville, MD
abdur@ir.iit.edu

David A. Grossman
Illinois Institute of Technology
Chicago, IL
dagr@ir.iit.edu

Ophir Frieder
Illinois Institute of Technology
Chicago, IL
ophir@ir.iit.edu

Abstract

In TREC-8, we participated in the automatic and manual tracks for category A as well as the small web track. This year, we focussed on improving our baseline and then introduced some experimental improvements. Our automatic runs used relevance feedback with a high-precision first pass to select terms and then a high-recall final pass. For manual runs, we used predefined concept lists focussing on phrases and proper nouns in the query. In the small web-track, we submitted one content-only run and two link-plus-content runs. We continued to use the relational model with unchanged SQL for retrieval. Our results show some promise for the use of automatic concepts, expansion within concepts and a high-precision first pass for relevance feedback.

1. Introduction

Our work for TREC-8 is a continuation of the work started in TREC-3 when we implemented an information retrieval system as an application of a relational database management system (RDBMS). We used unchanged Structured Query Language (SQL) to implement vector-space relevance ranking [Grossman95, Grossman96]. TREC-4 work demonstrated the relational implementation on category A data and introduced the concepts-list approach in the manual runs. In TREC-5, we implemented relevance feedback and entered the Spanish, Chinese and Confusion tracks. For TREC-6, we expanded our relevance feedback methodology to include the lnc-ltc term weights [Singhal96]. During TREC-6, we explored the assumption that certain infrequently occurring terms with high collection weights may actually be artificially inflating the query-to-document relevance ranking scores. We continued that work in TREC-7 with expanded stop lists and term thresholding. In addition, with TREC-7 we combined information extraction (IE) techniques with information retrieval through the use of a relevance feedback filter based on IE. During each of those years, our system performed well, but we noted that our baseline results were below those of other teams using similar retrieval strategies. So this year, we focused first on improving our baseline and then on experimentation with automated concepts and various expansion techniques, including a high-precision first-pass relevance feedback technique.

We began entering the manual track in TREC-4. This effort has focussed on structuring queries via concepts and manual relevance feedback while spending less than one half hour on each query. In TREC-5, we

experimented with the use of manually assigned term weights. For TREC-6, we used inexact term matching and an automatically generated thesaurus based on term co-occurrence. In TREC-7, our manual run focused on using phrases and proper nouns within the concepts. In addition, a more detailed iterative process was introduced. These manual techniques landed us among the top participants in manual track for TREC-7. This year, we continued the successful techniques and worked to ensure that we added key proper nouns and phrases for each concept in the query

We participated in the small web track introduced this year. Our relational platform proved to be quite flexible and was able to index the web documents with minor changes to the pre-processor (parser.) Our baseline (content-only) run used the straightforward vector space model with Singhal's pivoted cosine normalization [Singhal96]. Our experimental (link-plus-content) runs used link information to weight and rerank documents retrieved.

2. Prior Work in Relational IR

The implementation of an Information Retrieval (IR) system using the relational model hinges on the use of a relation (table) to model an inverted index, which is the central data structure in traditional IR systems. The traditional inverted index stores each unique term or phrase from the collection and a list of all the documents containing each term/phrase. The inverted index can also include frequency, offset, or other desired information. In the relational approach, this index is normalized and stored in a table. Queries using standard structure query language (SQL) are used to find and rank all documents containing the query terms. Full details of the implementation can be found in Grossman97 and Lundquist97. One benefit to using the relational model for IR is the ability to exploit parallel processing via the DBMS. All commercial DBMS systems offer a parallel version. For TREC-8, our manual runs used Windows NT versions of NCR/Teradata and Sybase/Adaptive Server Enterprise on Pentium SMP servers. This year's ad hoc and small web track submissions used Oracle on SUN Solaris machines.

3. Implementation Details

In this section we first discuss the baseline improvements made to our system and then we present our work in each track – automatic ad hoc, manual ad hoc and the small web track.

3.1. Improving the Baseline

In this year's work, we focused on the fundamentals and conducted many comparisons with the best systems from last year's TREC. The baseline title+description runs for the top three performers at TREC-7 were OKAPI 0.233, ATT 0.218 and UMASS 0.20. Our own baseline for TREC-7 queries was 0.17. We

looked for system differences to explain this lower performance. We began by examining the difference that the retrieval strategy makes. We implemented the same probabilistic retrieval strategy as given in [Robertson98]. We found that average precision recall did not differ significantly from previous runs using vector space strategies. We analyzed the result sets and found that they had very high overlap (relevant and nonrelevant) and very similar rankings. We concluded that the different retrieval strategies (when based on $tf*idf$) do not account for the differences in average precision recall.

We next considered the impact of token selection. Various stemming, phrases, and thesaurus techniques impact the tokens that represent the documents and the queries. We noted that the GSL file was instrumental in the OKAPI systems token selection -- conflating acronyms with their terms, American and British term variants, as well as many synonym groups. The GSL file only affected a few TREC-7 queries, but it had a large positive impact on almost all that it affected. In addition, the leading systems all used stemming approaches, while we did not. Phrase usage varied across the systems and was reported to result in a .1 to .2 improvement over terms, which is consistent with our own phrases. Stop lists also varied across the systems but it was unclear that this impacted precision/recall. We experimented in all of these areas, and found the keys improving our baseline were the 'stemming' and our title-phrase generation. We used the *kstem*+Porter equivalence groups developed at UMASS to add term variants to the query [Allan98]. This 'stemming' was quite effective and landed us at 0.196 average precision recall for title+description.

Our phrase generation technique uses every pair-wise combination of title terms. These new phrases helped (although not by much) most TREC-7 queries and did not cause serious degradation on any query. So we kept the technique for our TREC-8 runs. Finally, we had reached 0.20 and decided this was close enough (matching the third best) and moved on to query expansion.

3.2 Automatic Runs: High Precision Relevance Feedback with Automated Concepts

To ensure the top documents used for selecting expansion terms were relevant, we implemented a high-precision filter. This filter set up a concept for each title query word, used the Porter/*k-stem* algorithm to expand terms in each concept, and then required a document to contain at least one term from each concept. For example, the query 401, "foreign minorities, germany", results in three "concepts" created: 1) foreign, foreigner, foreigners; 2) minority, minorities 3) german, germany. The high precision first pass requires at least one word from each concept to be present for a document to qualify. Essentially, this is a logical AND of several OR groups. Ranking was achieved with the usual vector space similarity measure.

To select terms from these top documents, we used a modified Rocchio approach with the additional filter of requiring the term to occur in at least 2 of the top documents. We experimented with the number of top documents and number of terms to use and found that 10 terms from documents was best (see Tables 1 and 2.)

We note that similar work has been done earlier (most notably [Mitra98]) but our specific variations (automatic title concepts expanded with k-stems and $N > 1$) are new and effective.

Test	Average Precision
Using top 1 doc	.1609
Using top 2 docs	.2287
Using top 10 docs	.2359

Table 1. Calibration of Relevance Feedback using 10 Terms (TREC-7)

Test	Average Precision
No Feedback	.1966
Add 10 terms	.2359
Add 20 terms	.2065
Add 30 terms	.2057
Add 40 terms	.2057
Add 50 terms	.2100

Table 2. Calibration of High-Precision Relevance Feedback using 10 Documents (TREC-7)

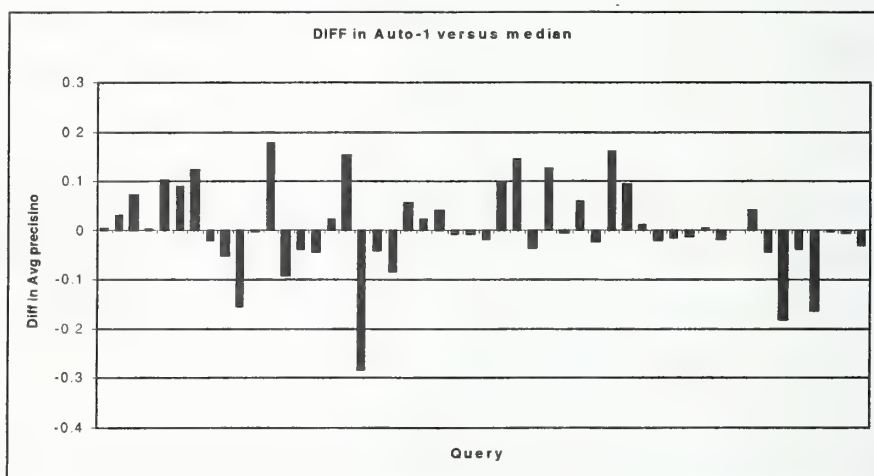


Figure 1. IIT Automatic Run-1 Difference from the Median

When we ran the second pass, we loosened the restriction of requiring at least one word from ALL title concepts to requiring at least one word from any ONE of the concepts. At this point our average precision recall was up to 0.2359. Finally, we reranked our resulting set of documents by the percentage of query terms found in the document. This reranking gained a small improvement, bringing our final TREC-7 run to 0.2454.

3.3. Manual Run

We spent approximately one-half hour formulating each query for our manual runs, using an iterative, interactive approach. The searcher used manual relevance feedback and general knowledge to identify new query words.

3.3.1. Manual Run Implementation Details

Consistent with previous years, our manual effort separated each query into a set of concepts -- search, scoring and negation. Search concepts are used in a vector space retrieval as a first pass. Terms from scoring concepts are then added to the document vectors and the documents are reranked. Finally, the negation concept 'disqualifies' a document from the result set. For TREC8, we used the negation concept more frequently than ever before—in 34 of the topics. Negation concepts included 147 phrases and 63 single words. Search concepts included 155 words and 498 phrases. The remaining tokens comprised the scoring-only concepts. The negative concept technique eliminated many irrelevant documents from our results. For example, on Topic 447 (*Stirling Engine*) we achieved 1.0 average precision recall by eliminated documents about *Stirling University* and people with the surname of *Stirling*.

	Search Concepts	Scoring Concepts	Negation Concepts	TOTAL
Terms	155	352	63	570
Phrases	498	567	147	1,212
Total	653	919	210	1,782

Table 3. Use of Concepts in Manual Track

As in our TREC-7 work, we emphasized phrases and proper nouns in the IIT manual ad hoc queries. For TREC-8, we used 1,782 search tokens including 1,212 phrases. Half of the phrases were proper nouns and the remaining were mostly common noun phrases. Of the 570 single words, 508 were either common or proper nouns. In other words, 96.5% of all search tokens were either phrases or single word nouns.

3.3.2. Manual Run Analysis

The average precision for our manual run was officially scored at 0.4104. We were at or above the median on 38 of 50 queries. When we were below the median, it was by a small margin and when we were above it was on average, by a much larger margin. We conducted failure analysis to determine why some queries performed poorly and why some did very well. This year, we spent more time reading the documents retrieved than any other year and believed most documents in the results sets to be relevant to the query. During our query development phase, the analyst tagged documents as relevant, doubtful, or non-relevant. We compared our list to the official results and found numerous differences (summarized in Table 4). Document relevance is subjective, of course, and subject to interpretation, but several of the differences in evaluation were difficult to reconcile. For example, Topic 423 asked for any references to Mirjana Markovic, the wife of Slobodon Milosevic – “Any mention of the Serbian president’s wife is relevant”. We found document FT942-3554 and FBIS3-2, both of which mention her by name and yet were judged non-relevant (see Figure 2).

NIST Relevance Assessment	IIT Relevance Assessment	Number of Documents
Relevant	Relevant	895
Relevant	Doubtful	188
Relevant	Non-Relevant	99
Non-Relevant	Relevant	428
Non-Relevant	Doubtful	356
Non-Relevant	Non-Relevant	1033

Table 4. Comparison of Relevance Judgments

Clearly such inaccuracies in relevance assessment have an impact on average precision recall. The average precision recall for our manual run increases to around .4800 when we use our relevance assessments.

<DOCNO> FT942-13554 </DOCNO>
taken from the text:
 "...Of special interest in Duga is the diary of **Mrs Mirjana Markovic, the wife of Mr Milosevic**. Her musings on the nature of life, spring-time in Belgrade often sound the death knell for the political rivals of her husband or herald an imminent Machiavellian manoeuvre by the Serbian President. The diary of Mrs Markovic is then reprinted in Politika, the oldest and most influential Serbian daily. . ."

<DOCNO> FBIS3-2 </DOCNO>
taken from the text:
 "... Independent biweekly that carries political and social commentary as well as articles focusing on popular culture. Regularly carries a column of political commentary written by **Mirjana Markovic--Milosevic's wife**—that often criticizes the Serbian nationalist cause. . ."

Figure 2. Sample Judgments for Topic 423

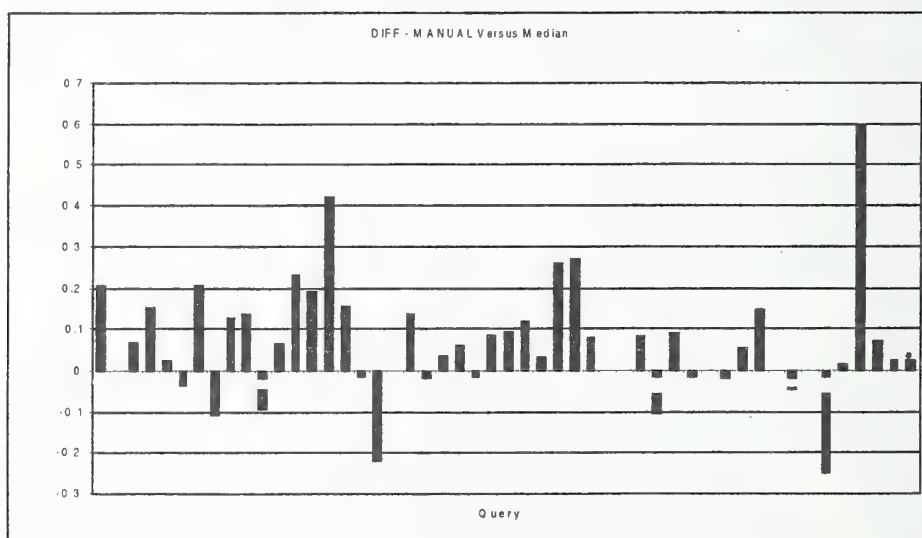


Figure 3. IIT Manual Run Difference from the Median

3.4. Small Web Track

This year we entered the new small web track. We used our baseline information retrieval system

with only minor changes to the preprocessor. In this section, we describe our techniques, results and analysis.

3.4.1. Small Web Track Implementation Details

Our Content-Only run (iit99wt1) simply used our baseline relational IR system to process the small web collection, using the title+description queries. The Link-Plus-Content runs (iit99wt2 and iit99wt3) began with the document sets retrieved during the Content-Only runs and then incorporated link data and reranked the results. Many of our initial efforts to incorporate links to or from other web pages resulted in reduced average precision values when measured against the TREC-7 benchmark data. We observed that the highest concentration of relevant retrieved documents occurred near the beginning of the documents retrieved for each topic; therefore, there was little or no need to reorder those high-ranking documents. We then retrieved documents beyond the original 1000 documents per query. We sought to use web links to identify and add documents to the result set. The approach we used was similar to the *root set* proposed in [Kleinberg97]. The top x documents (50 for Run-1, iit99wt2, and 100 for Run-2, iit99wt3) were included in the root set. The root set was then expanded so that links to and from those documents were added to the set of retrieved documents *if* they were already present in the set of all documents retrieved for a specific topic. In order to keep the result set within the maximum 1000 documents per topic, the lowest ranking documents from the original Content-Only run were removed from the result set. New documents were weighted and added to the retrieved documents set in such a manner that their original rankings were retained within the new result set.

Run Description	Relevant Retrieved	Average Precision
Content-Only	4480	0.2817
Link-Plus-Content	4523	0.2861

Table 5. IIT Small Web TREC-7 Benchmarks

3.4.2. Small Web Track Results

A comparison of results from our small web track runs is shown in Table 6. Our Content-Only run (iit99wt1) scored below the median on 27 of the 50 topics. We were neither the best nor the worst on any topic. When compared again against the median, our performance for Run-2 (iit99wt2, Link-Plus-Content) was greatly improved over the Content-Only run (iit99wt1). We received the best average precision score on three topics (419, 423, and 435) and were equal or above the median on 34 of the 50 queries. Since our average precision remained the same as the Content-Only run (at 0.2265), the relative improvement over the median is due to other teams degrading in their link-based runs.

Run Description	Run Identifier	Average Precision	Judged Relevant	Relevant Retrieved
Content-Only	iit99wt1	.2265	2279	1575
Link-Plus-Content (Run 1)	iit99wt2	.2265	2279	1572
Link-Plus-Content (Run 2)	iit99wt3	.2264	2279	1568

Table 6. IIT Small Web TREC-8 Results

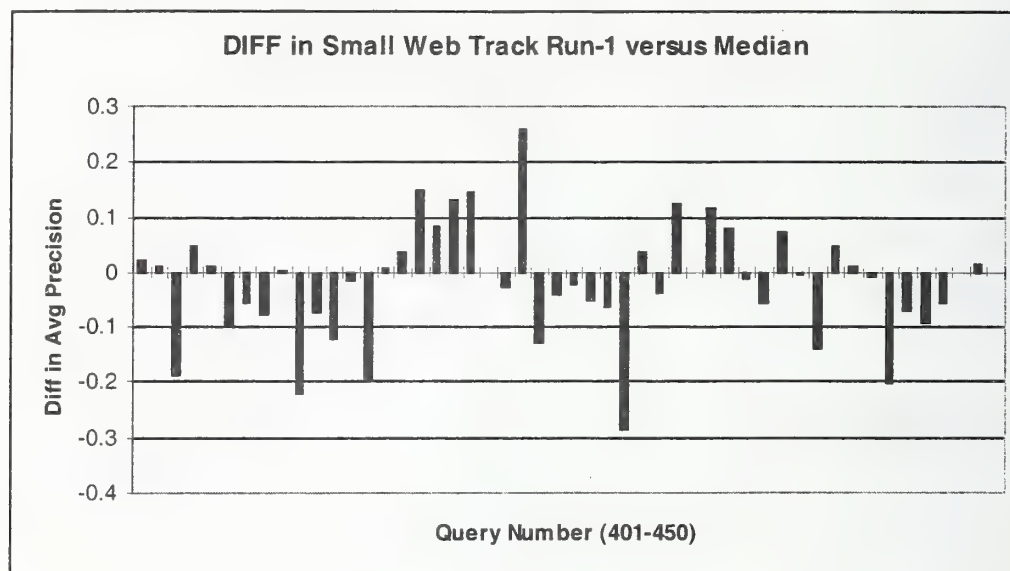


Figure 4. IIT Small Web Track Run-1 (Content-Only) Difference from the Median

3.4.3. Small Web Track Analysis

Incorporating link information is a challenging problem. As numerous studies have noted, all web links are not of equal value [Spertus97, Kleinberg97]. We have not yet found an effective way to automatically evaluate and discriminate between the numerous types of links that exist within web-based documents. Our excellent performance on query 423 can be attributed to the underlying retrieval engine and not to any specific techniques for web documents. We did well on it for the concept only run as well as for the link-based runs. The same can be said for our poor performance on query 403 and 429. An interesting factor in analyzing web track results is found in the sparseness of the qrels set. The TREC-8 qrels set is only 35 percent as large as the TREC-7 qrels set (2279 vs. 6495) and only 50 percent as large as the ad hoc track. (2279 vs. 4728).

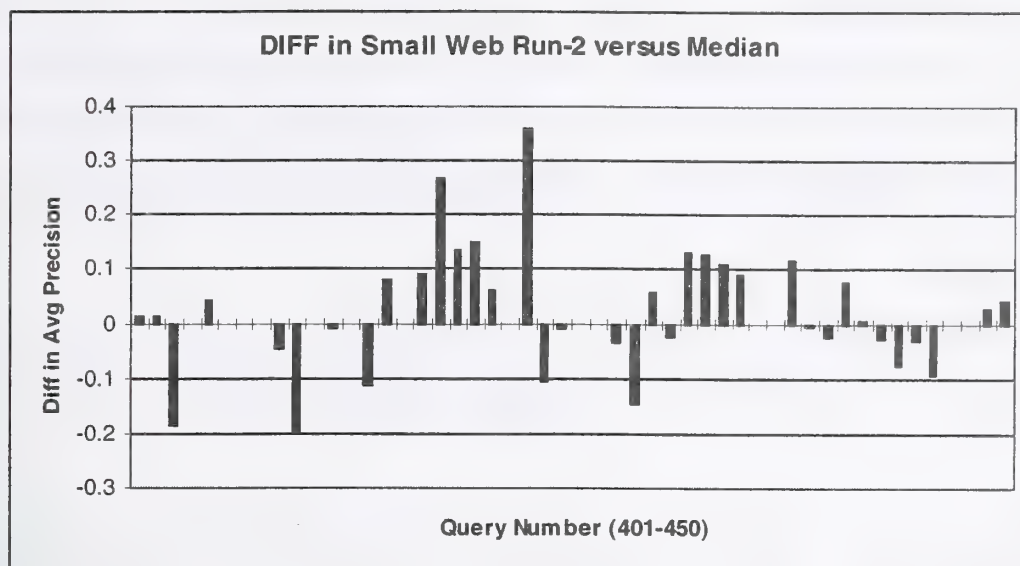


Figure 5. IIT Small Web Track Run-2 (Link-Plus-Content) Difference from the Median

4. Conclusions and Future Work

For TREC-8, we focused on improving our baseline system and then introducing some new feedback techniques. We identified key enhancements to our parser and our feedback engine. We introduced a technique for using k-stem conflation to expand title-term concepts and use this as a filter for high-precision relevance feedback. Our success in the manual track shows that phrases and nouns are important elements in runs with high average precision. Our work in the web track was a good beginning, but our results highlight the fact that there is still much room for improvement. Adjusting content runs based on link information assumes accurate content-only results and link information that can effectively weight and rank those results. Research will continue to improve both elements.

Table 7 summarizes the results of IIT TREC-8 submissions.

	iit99au1 (Tit+Des)	iit99au2 (Tit+Des)	iit99ma1 (Manual)	iit99wt1 (Content)	iit99wt2 (Link-Plus)
TREC-8 Track	Ad Hoc	Ad Hoc	Manual	Sm Web	Sm Web
Avg. Precision	0.2305	0.2041	0.4104	0.2265	0.2265
Precision at 10 Documents	0.4749	0.4343	0.7790	0.4100	0.4100
Documents Judged Relevant	4728	4728	4728	2279	2279
Relevant Retrieved	2688	2207	3106	1575	1572
At or Above Median (Avg. Prec.)	23	-	37	23	34
Below Median (Avg. Prec.)	27	-	13	27	16

Table 7. IIT TREC-8 Results Summary

Our future challenges include: (1) further integration of information extraction in relevance feedback, (2) the need to move beyond proper nouns and experiment with the use of entities as feedback filters, and (3) methods to more effectively evaluate and weight link information. In addition, the automation of the manual techniques used to add high quality phrases into our searches is an area for future work.

Acknowledgments

We wish to thank the director and staff at the Major Shared Resource Center, U.S. Army Research Lab, Aberdeen, MD for their generous support in making the small web track research possible.

References

- (Allan98) Allan, J.A., J. Callan, M. Sanderson, J. Xu, and S. Wegmann. "Inquiry and TREC-7". *Proceedings of the Seventh Text REtrieval Conference (TREC)*, sponsored by the National Institute of Standards and Technology and the Advanced Research Projects Agency, November 1998.
- (Buckley95) Buckley, C. A. Singhal, M. Mitra, and G. Salton, "New Retrieval Approaches Using SMART: TREC-4," *Proceedings of the Fourth Text REtrieval Conference (TREC)*, sponsored by the National Institute of Standards and Technology and the Advanced Research Projects Agency, November 1995.
- (Fox90) Fox, Christopher. A Stop List for General Text. *SIGIR Forum*, (v. 24, no. 1-2) 1990, p. 19-35.
- (Grossman95) Grossman, D., D. Holmes, O. Frieder, M. Nguyen, and C. Kingsbury, "Improving Accuracy and Run-Time Performance for TREC-4," *Proceedings of the Fourth Text REtrieval Conference (TREC)*, sponsored by the National Institute of Standards and Technology and the Advanced Research Projects Agency, November 1995.
- (Grossman96) Grossman, D., C. Lundquist, J. Reichert, D. Holmes, and O. Frieder, "Using Relevance Feedback within the Relational Model for TREC-5," *Proceedings of the Fifth Text REtrieval Conference (TREC)*, sponsored by the National Institute of Standards and Technology and the Advanced Research Projects Agency, November 1996.
- (Grossman97) Grossman, D., D. Holmes, O. Frieder, and D. Roberts, "Integrating Structured Data and Text: A Relational Approach," *Journal of the American Society of Information Science*, January 1997.
- (Kleinberg97) Kleinberg, Jon M. "Alternative Sources in a Hyperlinked Environment," *IBM Research Report (RJ-10076)*, May 29, 1997.
- (Lundquist97) Lundquist, C., D. Grossman, O. Frieder, and D. Holmes, "A Parallel Implementation of Relevance Feedback using the Relational Model," *Proceedings of the World Multiconference on Systemics, Cybernetics, and Informatics*, July 1997.
- (Lundquist98) Lundquist, C., D. Holmes D. Grossman O. Frieder. "Expanding relevance feedback in the relational model." *NIST Special Publication 500-240*, pages 489-502, August 1998.
- (Mitra98) Mitra, M., A. Singhal, C. Buckley. "Improving Automatic Query Expansion". *Proceedings of the Twenty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* ACM SIGIR'98 pages 206-214, 1998.
- (Robertson98) Robertson, S.E., S. Walker, M. Beaulieu. "Okapi at TREC-7: Automatic ad hoc, filtering, VLC and interactive track". *Proceedings of the Seventh Text REtrieval Conference (TREC)*, sponsored by the National Institute of Standards and Technology and the Advanced Research Projects Agency, November 1998.
- (Singhal96) Singhal, A., C. Buckley, and M. Mitra, "Pivoted Document Length Normalization," *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Ed. Hans-Peter Frei, Donna Harman, Peter Schauble and Ross Wilkinson, August 18-22, 1996.
- (Spertus 1997) Spertus, E. "ParaSite: Mining Structural Information on the Web," *HyperProceedings of the Sixth International World Wide Web Conference*. Electronic copy: <http://atlanta.cs.nichu.edu.tw/www/PAPER206.html>, 1997.

Automatic Query Feedback using Related Words

Stefan M Rüger
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, England
s.rueger@doc.ic.ac.uk

Abstract. Our experiments for the ad hoc task of TREC 8 were centered around the question how to create an automatic query feedback from the documents returned by an initial query.

1 The Query Process

1.1 Preprocessing of the Documents

In our retrieval experiments with $N = 528,155$ articles, we folded all words to lowercase and indexed all words with a document frequency of no more than 30% and which were not one of 349 stop words. For each document i we compute a weighted *document length* of

$$l_i = \sqrt{\sum_j (t_{ij} \log(0.3N/d_j))^2}, \quad (1)$$

where t_{ij} is the term frequency of word j in document i and d_j is the document frequency of word j .

Additionally, we identified roughly 100 *potentially interesting words* per document which we stored along with the meta-data of the document at index time. For these words we only kept nouns and adjectives based on Brill's tagger (Brill 1994) with a medium document frequency: the noun had to appear in least three documents and in no more than 30% of all documents. This resulted in a vocabulary of around 280,000 potentially interesting words. In our system we store a set of around 100 potentially interesting words per document along with the meta-data of the document at index time. Note that a set H of documents returned by a query may still have a potentially-interesting-words vocabulary of 10,000s of different words.

1.2 Processing of the Topics

We were only looking at the title and the description field of the topics, not at the narrative field. Each word of the title is included in the list of query words and also each non-stop word of the description field. For the description field, we had enhanced the stop words by about 60 function words of typical queries such as "relevant," "information" etc. Also, we decided to weigh the contribution of the words stemming from the description field with a factor of 1/5. A typical query would look like "blood-alcohol fatalities blood-alcohol:0.20 level:0.20 automobile:0.20 accident:0.20 fatalities:0.20".

1.3 First-Stage Document Retrieval

For each word j in the query list, we get the list of documents containing this word; we associate a score s_{ij} to each word j of document i in this list:

$$s_{ij} = q_j \cdot t_{ij} \cdot (\log(0.3N/d_j))^5, \quad (2)$$

where q_j is the query weight (here 1.0 for words from the title and 0.2 for words from the description). Words with a document frequency of more than $0.3N$ had not been indexed and are associated to the empty document list. The exponent 5 is slightly non-standard (one would expect a 2), but our experiments have shown a beneficial ranking behaviour wrt earlier TREC queries.

We get another list of documents by broadening the query word using Porter's stemming and the same weighing scheme as above (2), d_j being the document frequency of the stem j and t_{ij} being the term frequency of the words with the stem j ; we do not consider stems with a document frequency of more than $0.3N$. We repeat this process for all n query terms and, hence, arrive at $2n$ document lists. Subsequently, we create the union of these lists adding the scores of a particular document wrt all the query words and counting the number $n_i \leq 2n$ for each document, how many of the query terms and stems have been matched. We compute the final score of a document as

$$s_i = \left(\frac{n_i}{n}\right)^5 \left(\sum_{j \in \text{query}} s_{ij} + \sum_{j \in \text{stems}} s_{ij} \right) / l_i \quad (3)$$

The left factor rewards in a nonlinear way the documents that contain most of the query terms. Note that the score is normalized wrt the document length (1). The resulting document list is ordered according to the scores.

1.4 Related Words

The top-ranked 100 documents form a subset $H \subset D$ of the whole document set D . We suggest ranking the importance of each potentially interesting word j in the potentially-interesting-word vocabulary of H with a weight

$$w_j = \frac{h_j}{d_j} \cdot h_j \log(|H|/h_j),$$

where h_j is the number of documents in H containing the word j , and d_j is the document frequency of j wrt D . The second factor prefers medium matched-document frequency h_j , while the first factor prefers words that specifically occur in the matched documents. The highest-ranked words are meant to be related to the query. Indeed, we have something like "hardware", "software", "IBM" etc as the top-ranked words when querying for "computer". In the following, we use the top-ranked 50 words (according to w_j) which have a weight of not less than 2.5% of the maximum weight $\max_j(w_j)$. These words are called *related words*.

1.5 Automated Query Feedback

Another ranked document list is created from a related-words query, however, without stemming and with a slightly different ranking than (3):

$$s_i = \left(\sum_{j \in \text{related}} s_{ij} \right) / l_i \quad (4)$$

The scores of the related-words document list are normalized, such that the top score is 0.4. The scores of the resulting list of the first-stage document retrieval (see Subsection 1.3) are normalized so that the respective top score is 1.0. Now these two lists are joined, adding scores if necessary. This final list is sorted according to the joint score, cut off at 1000 and returned as result of the query.

2 TREC Evaluation and Conclusions

The ingredients to our information retrieval approach were ranked keyword retrieval using initial query words from the title, most words from the description of a topic and up to 50 related words wrt the results of an first-stage query. The above weighting and scoring schemes were picked using queries and relevant assessments of previous TRECs, though no systematic study has been carried out to optimize these schemes due to Black of time. During our preliminary studies, it was observed that query feedback improves precision and recall. Our scoring and ranking schemes seem rather ad hoc. We are convinced that a more systematic study could have improved the results considerably. One might want to lay out a more generic ranking scheme than the one above and introduce a reasonable number of parameters which are adjusted using training data and test data from previous TREC conferences and taking care of the potential pitfall of overfitting. This could lead to a potentially stronger ranking function, albeit one that is less applicable to theoretical reasoning. In addition, one might want to identify different classes of queries and appropriate ranking functions for each class to optimize the retrieval further. This is left to further studies.

References

Brill, E. (1994). Some advances in rule-based part of speech tagging. In *AAAI*.

Acknowledgements: This work was partially supported by the EPSRC, UK.

Two-Step Feature Selection and Neural Network Classification for the TREC-8 Routing

Mathieu Stricker ^{*,**}

Frantz Vichot ^{*}

Gérard Dreyfus ^{**}

Francis Wolinski ^{*}

* Informatique-CDC – Groupe Caisse des
Dépôts
Direction des Techniques Avancées
4 rue Berthollet
94114 Arcueil cedex - France
{forename.surname}@icdc.caissedesdepots.fr

** ESPCI
Laboratoire d'Electronique
10 rue Vauquelin
75005 Paris, France
{forename.surname}@espci.fr

1 Introduction

At the Caisse des Dépôts et Consignations (CDC), the Agence France-Presse (AFP) news releases are filtered continuously according to the users' interests. Once a user has specified a topic of interest, a filter is customized to fit this user's profile. Until now, these filters would rely on rule-based methods, whose efficiency is proven [Vichot *et al.*, 1999], but which require a large amount of work for each specific filter. This drawback can be avoided by using statistical methods which have the ability to learn from examples of relevant documents. Recently, we have developed a methodology for the AFP corpus. This paper presents its application to the TREC-8 corpus.

For the TREC-8 routing, one specific filter is built for each topic. Each filter is a classifier trained to recognize the documents that are relevant to the topic. When presented with a document, each classifier estimates the probability for the document to be relevant to the topic for which it has been trained. Since the procedure for building a filter is topic-independent, the system is fully automatic. Therefore, we describe it for one topic; the procedure is repeated 50 times.

By making use of a sample of documents that have previously been evaluated as relevant or not relevant to a particular topic, a term selection is performed, and a neural network is trained. Each document is represented by a vector of frequencies of a list of selected terms. This list depends on the topic to be filtered; it is constructed in two steps. The first step defines the characteristic words used in the relevant documents of the corpus; the second one chooses, among the previous list, the most discriminant ones. The length of the vector is optimized automatically for each topic. At the end of the term selection, a vector of typically 25 words is defined for the topic, so that each document which has to be processed is represented by a vector of term frequencies.

This vector is subsequently input to a classifier that is trained from the same sample. After training, the classifier estimates for each document of a test set its probability of being relevant; for submission to TREC, the top 1000 documents are ranked in order of decreasing relevance.

2 Problem and data description

In order to build the users' profile, routing participants were allowed to use the relevance judgments from the 1992 Financial Times (FT92) collection and any other relevance judgments from any other parts of the TREC collection except the 1993-94 Financial Times collection (FT93-94).

The FT93-94 collection is the test set; it contains 140,650 documents. Participants are asked to return a ranked list of the top 1000 retrieved documents from this part of the collection for each topic.

The routing profiles for TREC-8 are to be built for topics 351 to 400. Relevance judgments for these topics are provided on the FT92 collection. Since the test set is part of the FT collection, it would have been desirable to use only the FT92 collection for training. However, there are, on average, only 11 relevant documents per topic on the FT92 collection. Since this number is too low for estimating safely the parameters of a statistical model, the training set, in our experiments, includes all the relevant documents available for these topics on the whole TREC corpus (except FT93-94): the Federal Register 1994 collection (FR94), the Foreign Broadcast Information Service (FBIS) collection and the LA Times collection. With these additional collections, the average number of relevant documents per topic is increased to 71.6 (median 55.5).

3 Building of the training set

In order to build an efficient filter, the training set must be large and representative enough of the classes to be learned. A subset of relevant examples and a subset of irrelevant ones compose the training set. Since the number of relevant documents for each topic is generally low, all the available relevant documents from FT92, FBIS, LA and FR are selected.

For the subset of irrelevant documents, the number of candidates is huge. However, since the subset of relevant documents typically includes less than one hundred examples, the subset of irrelevant documents is limited to a few thousands. For each topic, two categories of irrelevant documents are available:

1. Those which have been judged irrelevant by a relevance assessor of TREC.
2. Those which have never been looked at; they are assumed to be irrelevant.

The documents from the first category have been checked because they were suspected to be relevant by some previous system (see TREC overview papers [Voorhees and Harman, 1999] for more details on the pooling technique used in TREC). Therefore, these documents can be said to be 'close' to the relevant documents: they are not representatives of the class of the irrelevant documents.

Some first experiments have shown that the results are better if we consider only irrelevant documents from the second category. Consequently, we sampled randomly 3,000 documents from this category only.

4 Term Selection

Each document of the FT92 collection is first tokenized into single words, case being ignored. In the following, each word is considered as a single unit called term. No stemming is performed.

We use no controlled vocabulary fields from the FT collection for building the filters.

The goal of the term selection is to define, for each topic, a vector of terms that will represent each document. The choice of these terms must be done very carefully since the quality of the filter relies heavily on this choice, whatever the model is. These terms must be chosen to allow a classifier to discriminate between relevant and irrelevant documents. The number of these terms is a tradeoff between two requirements: the larger the number of terms, the larger the number of examples required to have a good estimate of the classifier parameters; however, discarding terms leads to information loss.

Consequently a term selection must choose an appropriate number of appropriate terms for each topic.

4.1 Topic frequency analysis

The total frequency of each term of the FT92 collection is computed. This value is called corpus term frequency. For each term of a relevant document, we compute the document term frequency divided by the corpus term frequency. The terms of the document are then sorted by decreasing order according to this ratio. Therefore, very common words (stop words), which have a very high frequency in the whole corpus, are at the bottom of the list. The most specific words are at the top of the list, and the very rare words are ranked first.

The first half of the list is saved and the second half is discarded for each document. All the lists of all the relevant documents are merged and the frequencies of each single term in this list are computed.

A final sorting of the terms is performed, in order of decreasing frequencies, so that the rare words are at the bottom of the list.

At the end of this process we have a list of terms from which very common terms and very rare terms have been discarded; the remaining terms are representative of the specific vocabulary of the topic.

Heterogeneity of the training set

The subset of relevant documents in the training set arises from several collections, but the corpus term frequencies are computed on the FT92 collection. Consequently, some words are much less frequent on the FT92 collection than on other corpuses. For example, the word "california" tends to occur much more frequently on the LA Times collection than on the FT92 collection.

Consequently a short list of stop words was defined to take into account this heterogeneity. This list includes words like "california", "los", "angeles", A better solution might have been to compute the frequencies over the whole collection, but since the test set was only part of FT collection, the solution with the stop words list was adopted.

This first step defines the vocabulary specific to the topic, but the remaining terms are not necessarily the most discriminant ones and some of them may be highly correlated like "buenos" and "aires" for topic 351.

The goal of the second part of the term selection is to choose, amongst the previous words, the most discriminant ones in order to achieve a good classification task.

4.2 Gram-Schmidt orthogonalization

The Gram-Schmidt orthogonalization technique [Chen *et al.*, 1989] is used to rank the remaining terms in order of decreasing relevance to the output. The method can be described as follows.

We consider a model with Q candidate terms, and a training set containing N examples of documents whose relevance is known. The relevance of each document is considered as the desired output of the model: +1 for a relevant document and -1 for an irrelevant document. We denote by $x^i = {}^T[tf_1^i, tf_2^i, \dots, tf_N^i]$ the vector of term frequencies of term i in the different examples. We denote by y_p the N -vector of the outputs to be modeled. We consider the (N, Q) matrix $X = [x^1, x^2, \dots, x^Q]$. The model can be written as $y = X\theta$, where θ is the vector of the parameters of the model.

The first iteration of the procedure consists in finding the vector of terms which best explains the output, i.e. which has the smallest angle with the output vector in the N -dimensional space of observations. To this end, the following quantities are computed:

$$\cos^2(x^k, y_p) = \frac{(x^k y_p)^2}{(x^k x^k)(y_p y_p)}, \quad k = 1 \text{ to } Q$$

and the vector for which this quantity is largest is selected. In order to eliminate the part of the output which is explained by the first selected vector, all remaining candidate inputs, and the output vector, are projected onto the null subspace (of dimension $N-1$) of the selected term. In this subspace, the projected input vector that best explains the projected output is selected, and the $Q-2$ remaining vectors of terms are projected onto the null subspace of the first two ranked vectors. The procedure terminates when all Q input vectors are ranked. At the end of this procedure, a list of terms, ranked in order of decreasing relevance, is available.

This method applies only to models that are linear with respect to their parameters. This drawback can be circumvented by noting that an input, that is irrelevant for a model linear with respect to its parameters, is very likely to be irrelevant, irrespective of the model. Therefore, term selection is performed with Gram-Schmidt orthogonalization, and the selected terms are used as inputs to a neural network.

Once the terms are ranked, the pending question is that of deciding to what depth, within the list, the terms should be selected. This is achieved by introducing a "probe" term, which is a random variable, and by ranking this term, just as all other candidate terms, with the procedure described in the above section. The candidate terms that are ranked below the probe should be discarded. Actually, the rank of the probe term is a random variable; therefore, one has to compute the cumulative distribution function of this variable, and, in the spirit of hypothesis tests, one must choose a risk of selecting a term although it is less relevant than a random input (typically 1% or 5%). The computation of the probability for a probe to be more significant than one of the n terms selected after iteration n can be found in [Stoppiglia, 1997].

4.3 TREC-8 Result for term selection

The term selection method described in the previous section generates a vector of discriminant terms automatically for each topic. The dimension of this vector is determined by the probe defined above and thus its length is customized for each topic. The average length of the vector over all the topics is 25 terms, the maximum length is 40, and the minimum length is 10.

Figure 2 shows the final 10 top terms of topics 351, 352 and 375:

Topic 351	Topic 352	Topic 375
islands	channel	fusion
aires	rail	fuel
argentine	terminal	energy
carlos	link	science
argentina	kent	hydrogen
exploration	eurotunnel	produced
falkland	developments	reaction
falklands	tunnel	cold
menem	jobs	electric
exploitation	railway	laboratories

Figure 1: Final 10 top words of the term selection

5 Neural Networks

For each topic, the term frequencies of the vectors defined above are used as inputs of a neural network. Since the number of relevant examples per topic is low, the simplest architecture is chosen for the neural network i.e. a simple unit with a hyperbolic tangent function. Therefore, our classifier actually performs a linear separation.

Each filter is the function: $\tanh(\sum_{i=0}^N w_i x_i)$

Where N is the size of the input vector for the topic considered, w_i are the parameters to be estimated called weights and x_i is defined for $i > 0$ by:

$$x_i = \begin{cases} -1 & tf_i = 0 \\ \frac{tf_i}{\text{Log}(L)} & tf_i > 0 \end{cases}$$

where tf_i are the term frequencies of the terms selected by the term selection procedure. For each classifier $x_0 = 1$. L is the length of the text. By dividing each term frequency by the logarithm of the length, we take into account the fact that for longer text, the terms tends to appear more often.

The classifier is trained by minimizing the mean square difference between the desired output and the actual value of the classifier on the training set. The desired value is +1 if the document is relevant and -1 if the document is not relevant.

After training, each document of the test set (FT93-94 collection) is processed through the network and the output of the network is a number between [-1;+1]. The document are then sorted by decreasing order of output, and the top 1,000 documents are submitted for evaluation.

There are several techniques for minimizing the mean square error; one of the most efficient algorithm is the BFGS Quasi-Newton algorithm [Bishop, 1995]. However, since the number of relevant examples is small, the minimum of the cost function corresponds to large weights, so that the networks produce essentially a binary output: +1 or -1. In this case, the network has been overtrained, and its generalization ability is low. Consequently, in order to avoid the saturation of the tanh function, an early stopping procedure is used: the cost is minimized with a gradient descent procedure, and training is stopped after a few epochs.

In other experiments we have tried to use hidden neurons in the network architecture, but this did not improve the results.

6 Result on the FT93-94 collection

Our methodology has been applied to the topics 351 to 400 for the routing subtasks. The performances are measured by uninterpolated average precision.

The results produced by the trec_aval¹ package are listed below. The topics 359 and 369, which have no relevant document in FT93-94, are not taken into account.

Retrieved:	48000
Relevant:	1276
Relevant Retrieved:	1129
Non interpolated average precision :	31,99
R-precision	31,59

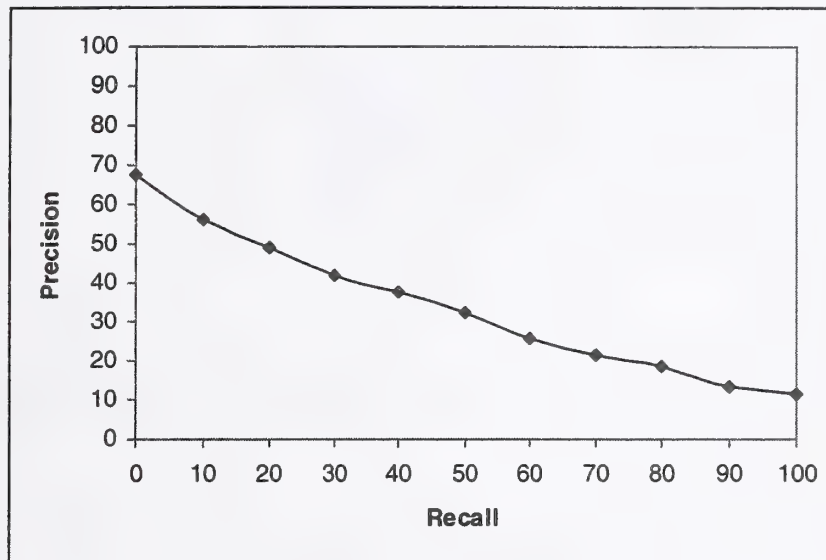
Interpolated Recall	Precision Averages
0	67,69
10	56,1
20	48,98

Precision at	
5 docs	40,00
10 docs	31,67
15 docs	27,50

¹ Available at <ftp://ftp.cs.cornell.edu/pub/smart/>

30	41,84
40	37,39
50	32,5
60	25,61
70	21,54
80	18,73
90	13,51
100	11,36

20	docs	24,90
30	docs	21,18
100	docs	12,27
200	docs	8,10
500	docs	4,19
1000	docs	2,35



7 Future Work

In this work, the terms used as inputs of the classifiers were actually single words. In the future, we plan to improve the text representation by using N-grams. These new terms can be handle in our method exactly like the single words. We expect the use of these N-grams to improve slightly the overall performance.

Our classifier was just a logistic regression due to the lack of training data available and the risk of overfitting. An early stopping procedure was used during the training process, but the number of epochs was not optimized for each topic. In some future experiments, we plan to use weight decay regularizers [MacKay, 1992] instead of the early stopping procedure to avoid overfitting.

REFERENCES

- [Bishop, 1995] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.
- [Chen *et al.*, 1989] S. Chen, S. A. Billings and W. Luo. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control*, Vol. 50, n°5, 1873 - 1896, 1989.
- [MacKay, 1992] D. J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, Vol. 4, n°3, 448-472, 1992.
- [Stoppiglia, 1997] H. Stoppiglia. Méthodes statistiques de sélection de modèles neuronaux; applications financières et bancaires. Thèse de l'université Paris VI, 1997.
<http://www.neurones.espci.fr/Francais.Docs/publications.html>
- [Vichot *et al.*, 1999] F. Vichot, F. Wolinski, H.-C. Ferri, D. Urbani. Feeding a Financial Decision Support System with Textual Information. *Journal of Intelligent and Robotic Systems*, Vol 26, 157 - 166, 1999

[Voorhees and Harman, 1999]. Ellen M. Voorhees and Donna Harman. Overview of the 7th Text Retrieval Conference (TREC-7). In *The Seventh Text Retrieval Conference (TREC-7)*, NIST Special Publication 500-242, 1999.

Mercure at trec8 : Adhoc, Web, CLIR and Filtering tasks

M. Boughanem, C. Julien, J. Mothe & C. Soule-Dupuy

IRIT/SIG
Campus Univ. Toulouse III
118, Route de Narbonne
F-31062 Toulouse Cedex 4
Email : trec@irit.fr

1 Summary

The tests performed for TREC8 were focused on automatic Adhoc, Web, Clir and Filtering (batch and routing) tasks. All the submitted runs were based on the Mercure system.

Automatic adhoc : Four runs were submitted. All these runs were based on automatic relevance back-propagation used in the previous TREC, with a slight change for one of these runs (Mer8Adtd3). A strategy based on predicting the relevance of documents using the past relevant documents was tested for this run. More precisely, instead of using the same relevance value for all top retrieved documents, some of them are selected and have their relevance value boosted.

Web : Four runs were submitted in this track:

1. content based only using Mercure simple search
2. content+ilink, according to Mercure architecture, we consider that document nodes are linked each other by weighted links. The top selected documents resulting from the initial search spread their signals towards the other document nodes. The documents were then sorted according to their activations, the top 1000 documents were submitted.
3. (2) + pseudo-relevance back-propagation method.
4. reranking of the 40 top documents using their links between each others.

Cross-language : Three runs were submitted for our first participation in this track. All these runs were based on query translation using an online machine translation . Two of these runs are a comparison between query translation from English to other languages and from French to other languages.

Filtering - batch and routing : The profiles were learned using three different strategies : Relevance Back-propagation (RB) and Gradient Back-propagation (GB) used in the

previous TREC and a new strategy based on Genetic Algorithm (GA). Four runs were submitted, two batch runs based on RB+GB and two routing runs, one based on RB+GB and the other one based on GA.

2 Mercure model

Mercure is an information retrieval system based on a connectionist approach and modelled by a multi-layered network. The network is composed of a query layer (set of query terms), a term layer representing the indexing terms and a document layer [3],[1].

Mercure includes the implementation of a retrieval process based on spreading activation forward and backward through the weighted links. Queries and documents can be used either as inputs or outputs. The links between two layers are symmetric and their weights are based on the tf.idf measure inspired from the OKAPI and SMART term weightings.

- the query-term (at stage s) links are weighted as follows:

$$q_{ui}^s = \frac{(1 + \log(tf_{ui})) * (\log(N/n_i))}{\sqrt{\sum_{i=1}^T (1 + \log(tf_{ui})) * (\log(N/n_i))^2}} \text{ if } tf_{ui} \neq 0$$

$$q_{ui} = 0 \text{ otherwise}$$

- the term-document link weights are expressed by:

$$w_{ij} = \frac{(1 + \log(tf_{ij})) * (h_1 + h_2 * \log(\frac{N}{n_i}))}{h_3 + h_4 * \frac{doclen_j}{avg_doclen}}$$

The query evaluation is based on spreading activation. Each node computes an input and spreads an output signal. The query modification is based on relevance back-propagation. It consists in spreading backward the document relevance from the document layer to the query layer [1].

2.1 Query evaluation

A query is evaluated using the spreading activation process described as follows :

1. The query u is the input of the network. Each node from the term layer computes an input value from this initial query: $In(t_i) = q_{ui}^s$ and then an activation value : $Out(t_i) = g(In(t_i))$ where g is the identity function.
2. These signals are propagated forwards through the network from the term layer to the document layer. Each document node computes an input : $In(d_j) = \sum_{i=1}^T Out(t_i) * w_{ij}$ and then an activation , $Out(d_j) = g(In(d_j))$.

The set of retrieved documents, $Output_u(Out(d_1), Out(d_2), \dots, Out(d_N))$ is then ranked in a decreasing order of the activation value.

2.1.1 Query modification based on relevance back-propagation

The top retrieved documents are judged and a *relevance value* is assigned to each document (positive for relevant documents, negative for non-relevant documents and nil for non-judged documents). These values are used to compute the *DesiredOutput* vector.

$DesiredOutput = (rel_1, rel_2, \dots, rel_N)$, $rel_j = \frac{Coef_rel}{Nb_rel}$ for relevant document
and $rel_j = \frac{Coef_rel}{Nb_Nrel}$ for nonrelevant document

1. This output is in fact considered as an "input" in the back-spreading process and is presented to the document layer. Each document node computes an input value, $In(d_j) = rel_j$ and then an activation signal, $Out(d_j) = g(In(d_j))$.
2. This activation is back-spread to the term layer. Each term node computes an input value, $In(t_i) = \sum_{j=1}^N (w_{ji} * Out(d_j))$ and then an output signal, $Out(t_i) = g(In(t_i))$.
3. Finally, the new query-term links corresponding to the new query are computed as follows:
 $Q_u^{(s+1)} = (q_{u1}^{(s+1)}, q_{u2}^{(s+1)}, \dots, q_{uT}^{(s+1)})$ with $q_{ui}^{(s+1)} = M_a * q_{ui}^{(s)} + M_b * Out(t_i)$

Notations :

T : the total number of indexing terms,

N : the total number of documents,

$q_{ui}^{(s)}$: the weight of the term t_i in the query u at the stage s ,

t_i : the term t_i ,

d_j : the document d_j ,

w_{ij} : the weight of the link between the term t_i and the document d_j ,

$doclen_j$: document length in words (without stop words),

avg_doclen : average document length, tf_{ij} : the term frequency of t_i in the document

D_j ,

n_i : the number of documents containing term t_i ,

M_a and M_b : tuned and determined by a series of experiments and set to $M_a = 2$ and $M_b = 0.75$.

3 Adhoc experiment and results

3.1 Adhoc methodology: Query modification based on past relevant documents

The main experiment undertaken this year concerns the use of past known relevant documents. The hypothesis that has been taken is : documents that have been judged as relevant for past queries could give a good approximation of relevance in pseudo-relevance feedback context.

Usually, the values used in pseudo-relevance back-propagation are : $coef_rel=1$ for the top 12 documents assumed as relevant and -0.75 for documents from 501-1000 assumed as non relevant.

The strategy tested this year consists of selecting some retrieved documents to be boosted instead of using the same relevance value for all the top retrieved documents. These documents are selected as follows:

1. Let us consider D_{ret} , the top 12 documents, $qrels_i$ the relevant documents for a given query q_i , i varying from 1 through 400 (past TREC qrels).

2. Build $comm_i = D_{ret} \cap qrels_i$, each $comm_i$ contains a set of documents from the top retrieved
3. Select the $comm_p$ containing the greatest number of documents and at least two documents.
4. Boost the coefficient value of the documents in $comm_p$. The coefficient value that has been used is 1.75.

3.2 Adhoc results and discussion

Four automatic runs have been submitted : Mer8Adtd1, Mer8Adtd2, Mer8Adtd4 (title+description) and Mer8Adtnd3 (long topic). These runs were based on a completely automatic processing of TREC queries and automatic query expansion based on pseudo feedback : Mer8Adtd4 is based on the query modification based on past relevant documents; Mer8Adtd1, Mer8Adtd2 and Mer8Adtnd3 are based on the classical relevance back-propagation using or not non-relevant documents.

TREC results			
Run	Best	\geq median	$<$ median
Mer8Adtd1	0	26	24
Mer8Adtd2	0	26	24
Mer8Adtnd3	0	28	22
Mer8Adtd4	1	27	23

Table 1: Comparative automatic adhoc results at average precision

Table 1 compares our runs against the published median runs. As mentioned in the previous TREC by [5], it is hard to tell much about relative performance since all automatic runs (title only, title+description and long topics) are in the same pool. Our results are in the average, with one best query.

Type	Run	average precision	Exact precision
title+description	basic td search	.2231	.2786
"	Mer8Adtd1 : 12 rel, 0 nrel	.2231 (0%)	.2786 (0%)
"	Mer8Adtd2 : 12 rel, 500 nrel	.2231 (0%)	.2786 (0%)
"	Mer8Adtd4 : 12 rel + past relevance	.2247 (1%)	.2733 (-2%)
Long Topic	basic tnd	.2327	.2928
"	Mer8Adtnd3	.2327 (0%)	.2931 (0%)

Table 2: Adhoc component results - 50 queries

Table 2 shows that the results obtained from all the tested strategies are quite the same. No strategy has improved the results. However, the results obtained with long topic seems to

be slightly better than with description+title only. The results obtained this year are quite the same than those obtained in our previous participation in TREC.

What did we gain from the strategy based on past queries (run Mer8Adtd4)? Note that : when the condition of boosting the relevance value in Mer8Adtd4 is not satisfied for a given query, the result of this query is the same than Mer8Adtd1. The difference between these runs can be seen only when some documents from the top selected are boosted.

topics	AvgP basic search	AvgP Mer8Adtd1	AvgP Mer8Adtd4
407	0.2120	0.3012 (42.08%)	0.2876 (35.66%)
411	0.1834	0.2497 (36.15%)	0.2497 (36.15%)
412	0.1565	0.2493 (59.30%)	0.3280 (109.58%)
416	0.2595	0.2540 (-2.12%)	0.2744 (5.74%)
422	0.2503	0.2896 (15.70%)	0.2896 (15.70%)
424	0.2279	0.0899 (-60.55%)	0.0955 (-58.10%)
427	0.1866	0.1776 (-4.82%)	0.1710 (-8.36%)
429	0.4561	0.2007 (-56%)	0.2621 (-42.53%)
431	0.4994	0.3061 (-38.71%)	0.3007 (-39.79%)
436	0.0313	0.0531 (69.65%)	0.0531 (69.65%)
438	0.2157	0.3007 (39.41%)	0.3119 (44.60%)
443	0.0912	0.0941 (3.18%)	0.0960 (5.26%)
446	0.1436	0.2548 (77.44%)	0.2548 (77.44%)

Table 3: Adhoc comparison between using the relevance of past documents and no

Table 3 compares the results between Mer8Adtd1 and Mer8Adtd4, query by query. It can be seen that 13 queries were concerned with the past relevance strategy. The average precisions of 7 of them were improved, 4 were unchanged and 3 were dropped.

4 Web Track

4.1 Web methodology

Mercure has been extended to take into account the links between document nodes. The document nodes in Mercure architecture are linked each others. The weight of the link between two nodes is computed as follows :

$$d_{ij} = \frac{||link(d_i) \cap link(d_j)||}{min(||link(d_i)||, ||link(d_j)||)}$$

where $link(d_i)$ is the list of documents (ilink or olink) linked to the document d_i . Only ilinks have been used in our experiments. Four runs using the title and the description fields, were submitted :

1. Mer8Wctd : content based only + pseudo-relevance feedback based on Mercure relevance back-propagation.
2. Mer8Wci1 : content+ilink. According to the spreading activation process of Mercure, the top 12 documents (resulting from the initial search) spread their signals through the document-document links towards the document nodes. Each document node computes an activation value as follows :

$$Out^{(new)}(d_j) = Out^{(old)}(d_j) + g\left(\sum_{k=1}^N d_{kj} * Out^{(old)}(d_k)\right)$$

The documents are then sorted according to their new activation. The top 1000 documents were then submitted.

3. Mer8Wci2 : (2) + pseudo-relevance back-propagation method. The top 12 retrieved documents in Mer8Wci1 were then used for relevance back-propagation and the top 1000 retrieved documents resulting from the new query were submitted.
4. Mer8Wci3 : rerank the 40 top documents using the links between documents. Instead of sorting the top 40 documents according to their activation value these documents are sorted according to another RSV (Retrieval Status Value) computed as follows : each document from the 40 top retrieved at the initial search computes an RSV :

$$RSV(d_j) = \sum_{d_k \in top\ 40\ doc.} d_{kj}$$

The 40 top documents are then sorted according to their RSV . The rank of the remaining documents 41-1000 is still unchanged.

Table 4 compares our runs against the published median results. Table (5) shows that using content only + relevance back-propagation (Mer8Wctd) gives better results than all the runs using the links. The strategy we used to take into account the links gives no improvement.

TREC results			
Run	Best	\geq median	$< median$
Mer8Wctd	0	23	27
Mer8Wci1	0	22	28
Mer8Wci2	5	19	31
Mer8Wci3	1	23	27

Table 4: Comparative automatic small web results at average precision

Type	Run	average precision	Exact precision
Title-Description	basic search	.1480	.1810
"	Mer8Wctd	.1638 (11%)	.1957 (8%)
"	Mer8Wci1	.1401 (-6%)	.1666 (-8%)
"	Mer8Wci2	.1488 (1%)	.1771 (-2%)
"	Mer8Wci3	.1435 (-3%)	.1741 (-2%)

Table 5: Small web component results - 50 queries

4.2 Comparison between Adhoc and Web runs

Table (6) compares Adhoc run (Mer8Adtd1) and Web run (Mer8Wctd). Both runs were based on Mercure pseudo-relevance back-propagation of the top 12 documents resulting from the initial Mercure search. The queries are built from the title and the description items. The results obtained using the Adhoc database are much better than those obtained using the Web database. A comparison with other participant runs will give some explanations for this difference.

Type	Run	average precision	Exact precision
T+D	Mer8Wctd : 12 rel, 0 nrel	.1638	.1957
T+D	Mer8Adtd1 : 12 rel, 0 nrel	.2231	.2786

Table 6: Comparison between adhoc and small web runs - 50 queries

5 Clir Experiment

This year is our first participation in cross-language track. It consists of three runs: two runs use English topics and retrieve documents from the pool of documents in all four languages (German, French, Italian and English), the other one uses French topics and retrieves documents from the pool of documents in all four languages. Our approach to the CLIR task is to translate the query from the source language into other languages using an online Machine Translation (MT). The runs were performed by doing individual runs for pair languages and merging the results to form the final ranked list. Before merging, the pseudo-relevance back-propagation method can be used. In this case, the top (12 in this experiment) documents resulting from the initial search by pair languages are assumed as relevant and thanks to the relevance backpropagation method, a new query is built for each language and applied to the index in the same language. The list of documents resulting from these searches are then merged to form the final ranked list (shows figure 1). The merging consists of sorting the documents resulting from the pair language search in decreasing order of their activation values. Four index files were created, one index by language. English words are stemmed using Porter, French words using the truncature (8 first characters), no stemming for Italian and German words.

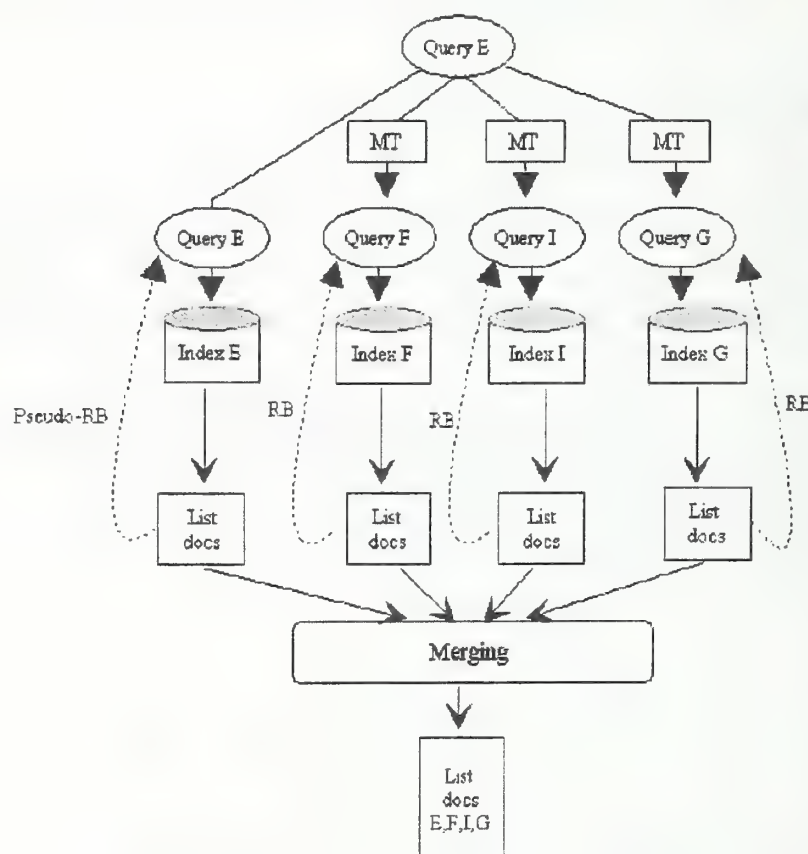


Figure 1: The cross-language methodology

Three runs were submitted.

- Mer8Can2x : query source in English + pseudo Relevance Backpropagation (RB)
- Mer8Cfr2x : query source in French + pseudo RB
- Mer8Can2x0 : query source in English

TREC results					
Run	Best	\geq median	$< median$	AvgP	R-prec
Mer8Can2x	4	10	18	0.2051	0.2386
Mer8Cfr2x	2	12	16	0.1620	0.2031
Mer8Can2x0	1	14	14	0.2059	0.2529

Table 7: Comparative automatic CLIR at average precision

Table 7 compares our runs against the published median runs. It can be seen that the run Mer8Can2x (English query source+RB) are better than Mer8Cfr2x (French query source+RB). The translation from English into the other languages seems to work better than from French. However, it can be seen that the pseudo relevance back-propagation does not bring anything (comparison between Mer8Can2x and Mer8Can2x0), the average precision even dropped.

6 Batch and Routing Experiments

The batch and routing experiments were performed using Mercure system. Three different learning techniques were tested to build the batch and the routing queries : the Relevance Back-propagation, the Gradient Backpropagation process presented in trec7 [3] and a new technique based on Genetic Algorithm (GA).

The FT92 documents were used as training data. The filtering algorithm starts with an initial query, built from all the parts of the topic, and its FT92 relevance judgments (positive and negative). The profiles were learned using the three techniques listed above. A pool of queries based on the three methods was then selected.

The GA we used to learn the profiles is not a classical GA. The GA operators we defined are based on some relevant information issued from the IR domain. The details of these operators can be found in [2].

6.1 Representation of IR for GA

The genetic algorithm attempts to make evolve, generation by generation, a population of queries towards those improving the outcome of the system. The competition starts between the various representations of the potential solutions the arbitrator of the competition is the fitness function.

6.1.1 Query individual and query population

The potential solution in our GA is a query. The initial set of queries (initial population) ($Pop^{(0)}$), contains the initial query and N given training documents. A subset of these documents can generate a different query. The query population is renewed at each iteration of the GA by applying selection, crossover and mutation operators.

6.1.2 Fitness function

A *fitness* is assigned to each query. This fitness represents the query effectiveness during the retrieving stage. It is computed according to the relevance of the retrieved documents. The formula used is:

$$Fitness(Q_u^{(s)}) = 1 - \frac{\sum_{i=1}^R \log(rank_i) - \sum_{i=1}^N \log(i)}{\log\left(\frac{(R!)}{(N-R)!R!}\right)} \quad (1)$$

where $Q_u^{(s)}$ is the query u at the generation (s) of the GA, $\|D_r\|$ is the number of relevant documents retrieved through all generations, $\|D_{nr}\|$ the number of nonrelevant documents retrieved through all generations.

6.1.3 Genetic operators

Some genetic operators defined in this work are not pure genetic operators. They have been adapted to take advantage of some techniques developed in IR. For this reason, the defined operators are qualified as knowledge based operators.

Selection.

The selection procedure is based on the roulette wheel selection [6]. It consists in assigning to every individual of the population a number of copies in the next generation, proportional to the individual relative fitness. Then, one spin of the wheel selects a single individual each time until the population size is reached.

Crossover based on term weight

This crossover does not use a crossing point, it modifies the term weights according to their distribution in the relevant and nonrelevant documents. Let us consider $Q_u^{(s)}$ and $Q_v^{(s)}$ two individuals (queries) selected for crossover. The result is the new individual $Q_p^{(s)}$ defined as follows :

$$\left. \begin{array}{l} Q_u^{(s)}(q_{u1}^{(s)}, q_{u2}^{(s)}, \dots, q_{uT}^{(s)}) \\ Q_v^{(s)}(q_{v1}^{(s)}, q_{v2}^{(s)}, \dots, q_{vT}^{(s)}) \end{array} \right\} \rightarrow Q_p^{(s+1)}(q_{p1}^{(s+1)}, q_{p2}^{(s+1)}, \dots, q_{pT}^{(s+1)})$$

$$q_{pi}^{(s)} = \begin{cases} \text{Max}(q_{ui}^{(s)}, q_{vi}^{(s)}) & \text{if } \text{weight}(t_i, D_r^{(s)}) \geq \text{weight}(t_i, D_{nr}^{(s)}) \\ \text{Min}(q_{ui}^{(s)}, q_{vi}^{(s)}) & \text{otherwise} \end{cases}$$

Where the weight of term t_i in a set of documents D is defined by $\text{weight}(t_i, D) = \sum_{d_j \in D} w_{ji}$ and $D_r^{(s)}$ is the set of relevant documents retrieved by the population $Pop^{(s)}$ and $D_{nr}^{(s)}$ is set of nonrelevant documents retrieved by the $Pop^{(s)}$.

Mutation based on term relevance

The mutation operator explores the terms occurring in the relevant documents in order to adjust the corresponding gene values in the query selected for the mutation.

Let us consider $Q_u^{(s)}$ as the selected individual query and $L^{(s)}$ as the set of terms from $D_r^{(s)}$, the relevant documents retrieved in the previous generation of the GA. The mutation alters the genes of the selected individual on the basis of the $L^{(s)}$ terms and on the probability P_m . The $L^{(s)}$ terms are sorted according to a score value calculated as follows:

$$\text{Score}(t_i) = \frac{\sum_{d_j \in D_r^{(s)}} w_{ji}}{\|D_r^{(s)}\|}$$

This score is used to limit the number of terms that can be used for the mutation process. The mutation operation is done as follows :

```

for each term  $t_i$  in  $L^{(s)}$ 
  if ( $\text{random}(p) < P_m$ ) then    /* mutate the gene  $t_i$ 
     $q_{ui}^{(s)} = \text{average}(Q_i^{(s)}) - \delta$   /* modify the weight of  $t_i$  in  $Q_u^{(s)}$ 
  endif
endfor

```

Where P_m is the mutation probability, $random(p)$ generates a random number p in the range of $[0..1]$. The average function is computed as follows :

$$average(Q_u^{(s)}) = \frac{\sum_j^T q_{ui}^{(s)}}{n_{q_{ui}}^{(s)}}$$

where $n_{q_{ui}}^{(s)}$ is number of $q_{ui}^{(s)} \neq 0$ in $Q_u^{(s)}$. δ is a parameter used to control the average value (we used $\delta = 0$).

Learning algorithm

- Submit the initial query and do the search
- Build the initial population as follows : the top 10 relevant documents join the initial query in the set of initial individuals
- Compute the fitness of these individuals
- Apply the genetic operators to these individuals as follows :
 - Repeat
 - Select two individuals, crossover then mutation
 - Until $Size(Pop^{(s+1)}) = fixed\ population\ size$ (the population is then built).
- Repeat
 - For each query in the population do a search
 - Compute the fitness of each query
 - Apply the genetic operators
- Until a fixed number of iteration

6.2 Routing task

For the routing task, the queries having the best average precision in the training data were selected as routing queries. Two runs were submitted : Mer8R1 based on relevance backpropagation and Mer8R2 based on GA.

TREC Routing				
Run	Best	$\geq median$	$< median$	AvgP
Mer8R1	3	14	34	0.271
Mer8R2	1	3	45	0.108

Table 8: Comparative routing results at average precision

Table 8 shows the comparative routing results at average precision. There was no gain from the GA-based run. The results obtained in TREC7 filtering database (AP collection) were much better than those obtained in the FT collection. This is probably due to this collection. In fact, a lot of queries has only few relevant documents in the part of the FT used as training data.

6.3 Batch Filtering

The profiles in the batch task are learned using the relevance backpropagation method and the gradient back-propagation. The pool of the queries contains the queries generated by both methods. The TREC standard output file of each query was analyzed to build an output file containing:

< topic > < func > < value > < thresh > < rank > < prec > < recall > < method >

As it has been done in [7]. The weights, corresponding to the document activation, which maximize the utility function were then found and selected as thresholds. Thus, the queries corresponding to these thresholds were selected and tested against the test data. The *< method >* field correspond to the used method (relevance backpropagation or gradient backpropagation).

Two runs were submitted : Mer8BaLF1 based on utility-[*LF1*] and Mer8BaLF2 utility-[*LF2*].

TREC batch filtering					
Run	Best (with score=0)	\geq median	$<$ median	$min = med = max = 0$	score = 0
Mer8BaLF1	17 (9)	38	7	5	22
Mer8BaLF2	15 (7)	33	10	7	23

Table 9: Comparative batch filtering results

Table 9 lists the comparative batch results. It can be seen that in both Mer8BaLF1 and Mer8BaLF2 runs the results are quite good, 17 best queries in the first run and 15 in the second run. It can be also noticed that 9 queries among the 17 best in the first run and 7 among the 15 best in the second run have a *score = 0*. It can be shown also that there are 22 queries with score zero (no documents have been submitted for these queries).

References

- [1] M. BOUGHANEM, C. CHRISMENT & C. SOULE-DUPUY, *Query modification based on relevance backpropagation in Adhoc environment*, INFORMATION PROCESSING AND MANAGEMENT. APRIL 1999.
- [2] M. BOUGHANEM, C. CHRISMENT & L. TAMINE, *Query space exploration based on GA*, INFORMATION RETRIEVAL JOURNAL. OCTOBER 1999.
- [3] M. BOUGHANEM, T. DKAKI, J. MOTHE & C. SOULE-DUPUY, *Mercure at trec7*, PROCEEDINGS OF THE 7TH INTERNATIONAL CONFERENCE ON TEXT RETRIEVAL TREC7, E. M. VOORHEES AND HARMAN D.K. (ED.), NIST SP 500-236, Nov. 1997.
- [4] C. BUCKLEY & AL, *Query zoning : TREC'5*, PROCEEDINGS OF THE 5TH INTERNATIONAL CONFERENCE ON TEXT RETRIEVAL TREC5, HARMAN D.K. (ED.), NIST SP 500-236, Nov. 1996.
- [5] C. BUCKLEY & AL, *SMART High Precision : TREC 7*, PROCEEDINGS OF THE 7TH INTERNATIONAL CONFERENCE ON TEXT RETRIEVAL TREC 7, E. M. VOORHEES AND HARMAN D.K. (ED.), NIST SP 500-236, Nov. 1997.

- [6] D. E. GOLDBERG *Algorithmes génétiques. Exploration, optimisation et apprentissage automatique* EDITION ADDISON-WESLEY, FRANCE 1994.
- [7] S. ROBERTSON AND AL *Okapi at TREC-6*, PROCEEDINGS OF THE 6TH INTERNATIONAL CONFERENCE ON TEXT RETRIEVAL TREC6, HARMAN D.K. (ED.), NIST SP 500-236, Nov. 1997.

The JHU/APL HAIRCUT System at TREC-8

James Mayfield, Paul McNamee and Christine Piatko
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723-6099 USA
James.Mayfield@jhuapl.edu
Paul.McNamee@jhuapl.edu
Christine.Piatko@jhuapl.edu

Goals

The Johns Hopkins University Applied Physics Laboratory (JHU/APL) is a second-time entrant in the TREC Category A evaluation. The focus of our information retrieval research this year has been on the relative value of and interaction among multiple term types and multiple similarity metrics. In particular, we are interested in examining words and n-grams as indexing terms, and vector models and hidden Markov models as similarity metrics.

Approach

The Hopkins Automated Information Retriever for Combing Unstructured Text (HAIRCUT) system was built to explore the use of multiple term types, including words, stemmed words, multi-word phrases and character n-grams of various lengths. The system is implemented in Java for ease of development and portability. It supports both a vector model and a hidden Markov model (HMM) for comparing queries against documents.

Under the vector model, terms are usually weighted by TF/IDF. Okapi BM 25 [Walker *et al.*, 1998] and plain TF weightings are also supported. Cosines can be computed either relative to the origin, or relative to the corpus centroid. Terms that appear in a high percentage of documents are stop-listed. The HAIRCUT HMM is a simple two-state model capturing both document and collection statistics [Miller *et al.*, 1999]. HAIRCUT also supports breaking documents into passages, although to date we have received no significant boost in average precision from doing so.

HAIRCUT performs rudimentary preprocessing on queries to remove stop structure [Allan *et al.*, 1998], e.g., affixes such as "... would be relevant" or "relevant documents should...". Other than this preprocessing, queries are parsed in the same fashion as are documents in the collection.

After the query is parsed and appropriately weighted (TF/IDF for the vector model, TF for the HMM; query section term-weighting was not used), an initial retrieval is performed with a single round of blind relevance feedback. We found a 27.6% relative increase in average precision on the TREC-7 ad hoc task when using relevance feedback. To perform relevance feedback, HAIRCUT first retrieves the top 1000 documents. The top 20 documents are then used for positive feedback, and the bottom 75 documents for negative feedback. Query terms are reweighted using the Rocchio score ($\alpha=3$, $\beta=2$, $\gamma=2$) and an affinity score, which is a function of the term's frequency in the retrieved documents and its frequency in the collection as a whole. The top-scoring terms, ignoring very high and very low frequency terms, are then used as the revised query.

After retrieval using this expanded and reweighted query, we have found a slight improvement by penalizing document scores for documents missing many query terms. We multiply document scores by a penalty factor:

$$PF = 1.0 - \left(\frac{\# \text{ of missing terms}}{\text{number of terms in query}} \right)^{1.25}$$

We use only about one-fifth of the terms of the expanded query for this penalty function:

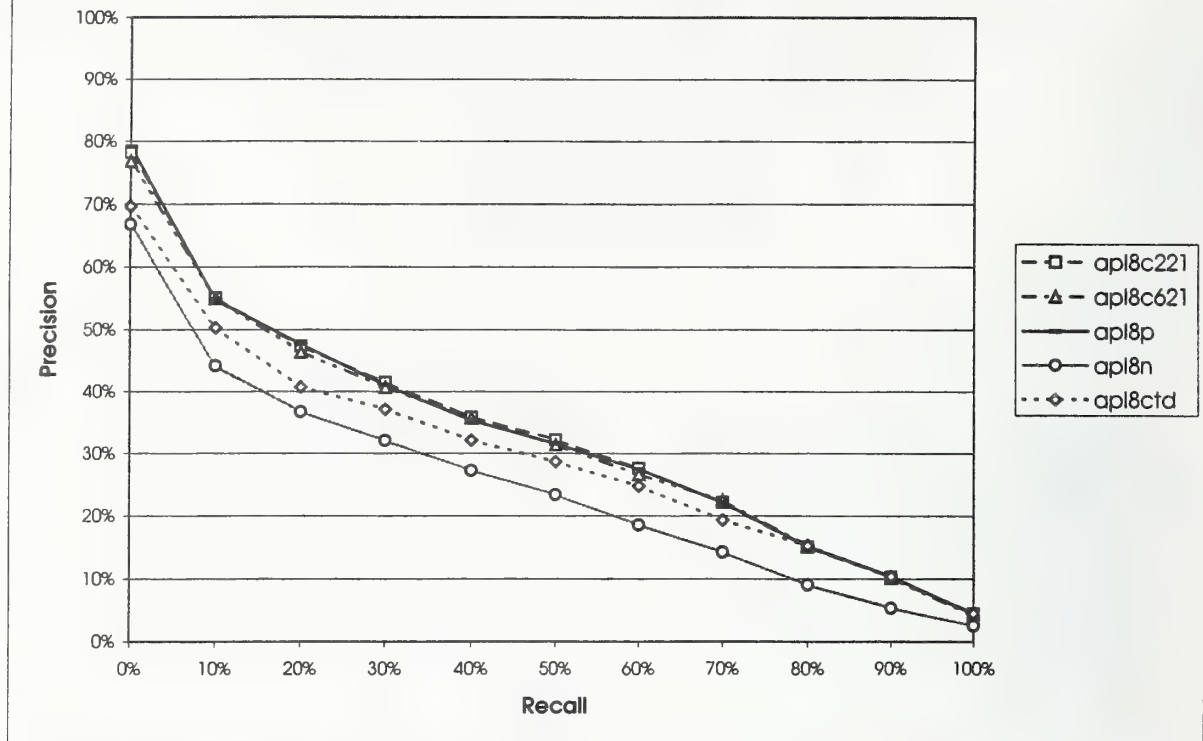
	# Top Terms	# Penalty Terms
words	60	12
6-grams	400	75

We conducted our work on a shared 4-node Sun Microsystems Ultra Enterprise 450 server. The workstation had 2 GB of physical memory and access to 50 GB of hard disk space.

The HAIRCUT system comprises approximately 25,000 lines of Java code.

For TREC-8 we tested three types of terms: stemmed words, 6-grams and phrases. After eliminating

Figure 1. JHU/APL TREC-8 Ad Hoc Results



punctuation, downcasing letters, and mapping numbers to a single digit, a word was any remaining blank-delimited sequence of characters. For n-grams we used 6-grams formed from the same character stream used for selecting words. In TREC-7 we used

5-grams [Mayfield & McNamee 1999], but we have found 6-grams to be preferable for English. Our use of 6-grams for other languages is based on convenience, not proven superiority.

Figure 2. APL TREC-8 Ad Hoc Results
Three Baseline Runs and One Merged Run

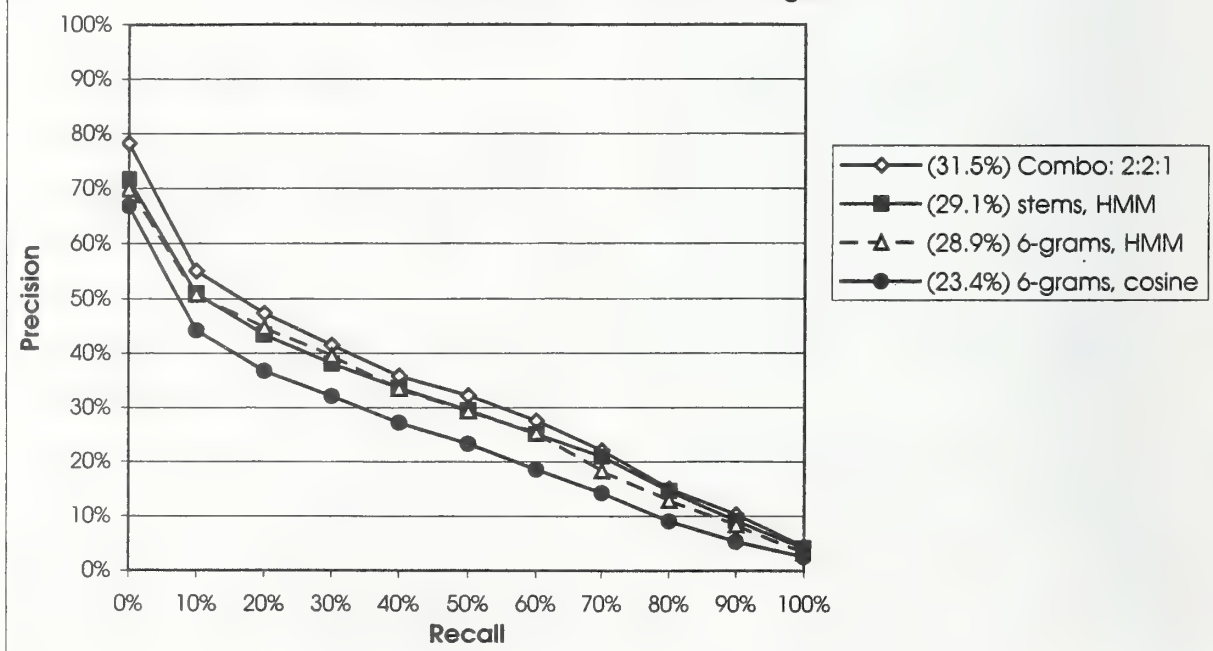
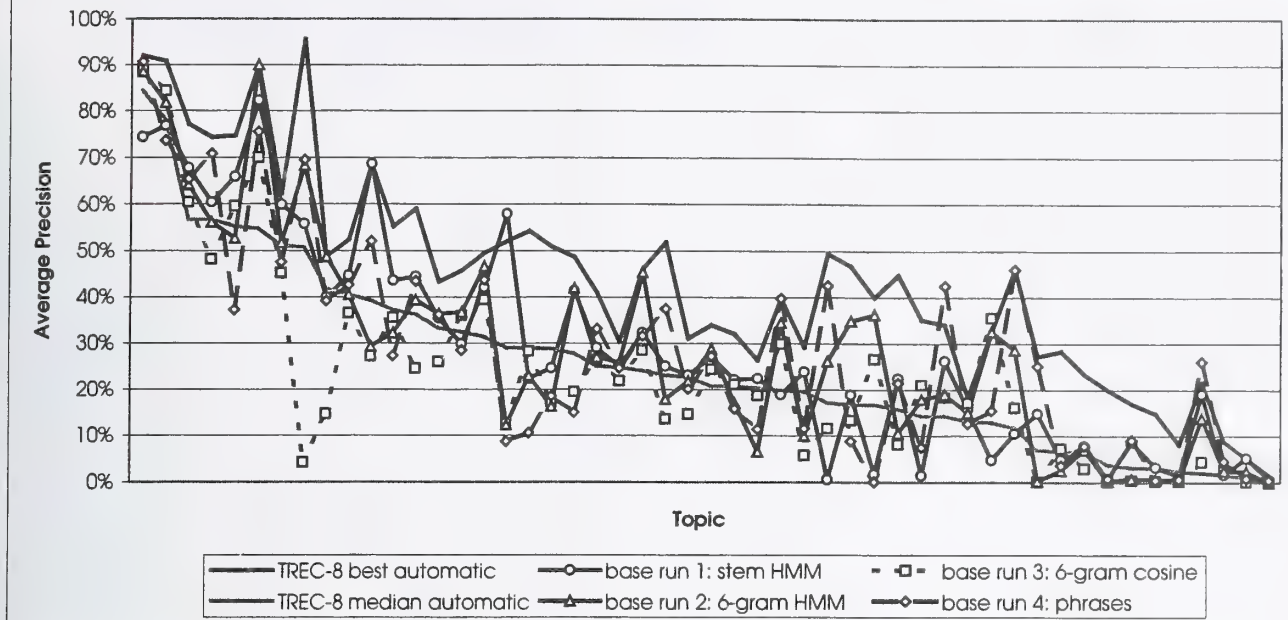


Figure 3. JHU/APL TREC-8 Ad Hoc Base Runs by Topic



Our focus this year was on combining evidence from multiple runs. We varied three system features to obtain runs that were then merged:

- use of a vector model versus use of a hidden Markov model;
- use of n-grams as terms versus use of words or stemmed words as terms; and
- use of phrases.

We used two different methods for merging these runs. In most cases we used linear combination of normalized scores. Scores were normalized simply by scaling the range of scores exhibited by the top 1000 documents to the range 0..1. In one case, we selected the retrieval technique on a per-query basis.

Ad hoc Results

The JHU/APL submissions were based on four underlying retrieval runs:

- **stemhmm**: an HMM run using stemmed words, with $\alpha=0.3$.

- **sixhmm**: an HMM run using 6-grams, using $\alpha=0.15$.

- **sixcos**: a vector-based run using 6-grams as terms and TF/IDF term weighting.

- **phrase**: an HMM run using common stem bi-grams, in addition to individual stems, as terms. Our phrase list comprised one million stem pairs.

JHU/APL submitted five ad hoc runs. All underlying runs were first normalized as described above. Four runs were based on topic/description/narrative (TDN); one run, as required by the TREC-8 rules, was based on topic and description (TD) only:

- **apl8c221**: A linear combination of stemhmm, sixhmm, and sixcos, with scores from both stemhmm and sixhmm receiving twice the weight of sixcos.
- **apl8c621**: A linear combination comprising six parts stemhmm, two parts sixhmm, and one part sixcos.

Figure 4. JHU/APL TREC-8 Official Ad Hoc Runs by Topic

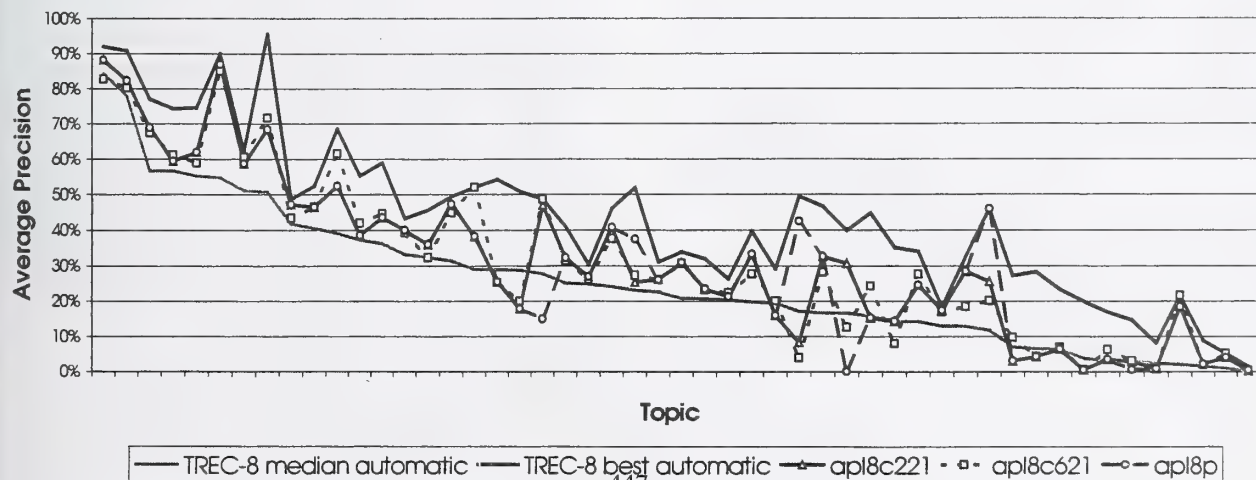
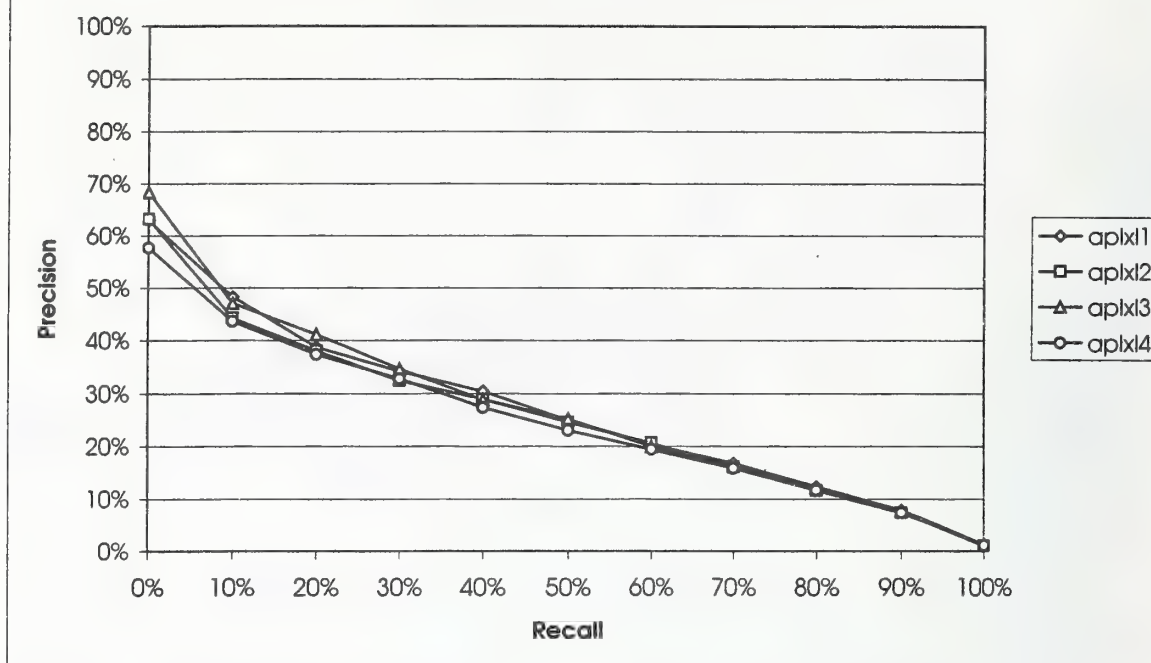


Figure 5. JHU/APL TREC-8 CLIR Performance



- **apl8p**: A query-by-query combination of apl8c221 and phrase, in which queries that the system judged likely to benefit from phrases were handled exclusively by the phrase run, while the remaining queries were handled exactly as in apl8c221.
- **apl8n**: sixhmm unmodified, submitted as a test of the efficacy of raw n-grams.
- **apl8ctd**: A TD run combining stemhmm, sixhmm, and sixcos in a 2:2:1 ratio.

Our official results for these five runs are shown in Figure 1. Aggregate numbers for the base and submitted runs are as follows:

	Average precision	Relevant retrieved	Precision @100
stemhmm	0.2914	3156	0.2378
sixhmm	0.2885	3061	0.2436
sixcos	0.2340	2919	0.2106
phrase	0.2850	3153	0.2410
apl8c221	0.3150	3332	0.2558
apl8c621	0.3126	3335	0.2558
apl8p	0.3154	3295	0.2568
apl8n	0.2885	3061	0.2436
apl8ctd	0.2860	3117	0.2324

Our submitted TDN runs exhibited eight of the top scores over the fifty queries; our single TD run exhibited four bests:

	Task pool	Best	At or above median
apl8c221	automatic TDN	0	40
apl8c621	automatic TDN	4	39
apl8p	automatic TDN	1	39
apl8n	automatic TDN	3	30
apl8ctd	automatic TD	4	37

Figure 2 shows the aggregate improvement obtained by combining three base runs to form apl8c221. Figure 3 shows the average precision for each of the fifty TREC-8 ad hoc topics produced by our four base runs. The chart shows wide variability in the responsiveness of queries to the four techniques. As shown in Figure 4, our linear combination method manages to outperform each of the base methods in aggregate by performing in the upper half of the range of base scores for most queries. Our phrase combination method, also shown in Figure 4, is a first attempt to select a retrieval method on a query-by-query basis.

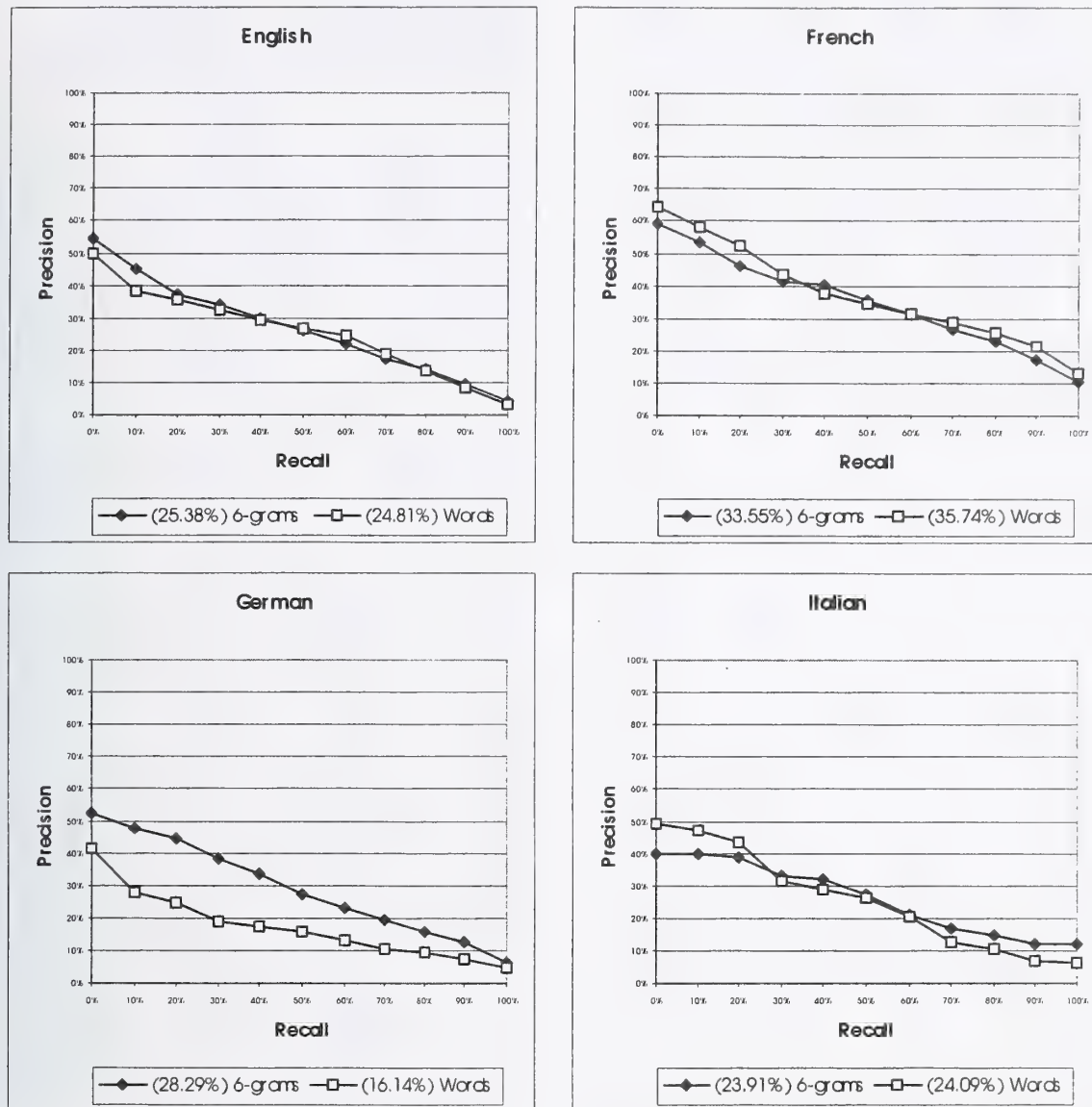
Cross Language Track

JHU/APL was a first-time entrant in the CLIR track at TREC-8. We used SYSTRAN[®] to translate English queries into German, French and Italian, HAIRCUT to perform both word-based and n-gram-based retrieval on the four collections, and linear combination of normalized scores to combine the eight runs.

We found that different languages responded differently to words and n-grams; thus (except for run aplx12) *within each language* we used the following weights to combine the two types of runs (which weights were derived from training on the TREC-7 CLIR task):

	English	French	German	Italian
6-grams	1.0	3.3	3.5	1.0
Words	3.0	1.0	1.0	1.2

Figure 6. Words and n-gram base runs for the four CLIR languages.



We submitted four CLIR runs:

1. **aplx11**: A combination that weighted the languages as follows:

English	French	German	Italian
2.3	2.0	2.0	1.8

These weights were also derived from training on the TREC-7 CLIR task. It is unclear whether these weights optimize average precision on the TREC-7 task because of differing HAIRCUT performance on the various languages, because of the number of relevant documents in the various languages, or because of other factors.

2. **aplx12**: A two-phase combination, in which the languages were first individually combined using the following weights:

	English	French	German	Italian
6-grams	1.0	2.5	8.5	1.0
Words	6.0	1.0	1.0	1.5

The resulting combined runs were then merged using these weights:

English	French	German	Italian
2.0	1.1	1.1	0.6

As with apxl1, these weights were trained from the TREC-7 CLIR task.

3. **apxl3**: A combination that weighted the four languages evenly:

English	French	German	Italian
1.0	1.0	1.0	1.0

4. **apxl4**: A combination that weighted English as 1.5, and the other languages as 1.0:

<i>English</i>	<i>French</i>	<i>German</i>	<i>Italian</i>
1.5	1.0	1.0	1.0

Results from these four runs are shown in Figure 5. The eight base runs are shown in Figure 6. Aggregate numbers for our submitted CLIR runs are as follows:

	<i>Average precision</i>	<i>Relevant retrieved</i>	<i>Precision @100</i>
aplx11	0.2571	1911	0.2714
aplx12	0.2471	1944	0.2782
aplx13	0.2542	1890	0.2711
aplx14	0.2389	1902	0.2575

The similarity of these runs lead us to believe that tuning weights under this approach to merging is not a cost-effective use of one's time.

Our monolingual results for the four languages, using the human-translated queries provided by NIST, were significantly below those seen on the TREC-7 CLIR task:

	<i>English</i>	<i>French</i>	<i>German</i>	<i>Italian</i>
T8 Words	0.2481	0.3780	0.2525	0.3152
T8 6-grams	0.2538	0.3342	0.3933	0.2778
T7 Words	0.4533	0.3715	0.3671	0.3936
T7 6-grams	0.4363	0.3767	0.4143	0.3281

The reasons for this drop in performance are unclear. Given HAIRCUT's average precision of 31.5% on the TREC-8 ad hoc task, it seems unlikely that the drop in English performance is due to some sort of overtraining on the TREC-7 data. We note a significant drop in the number of relevant documents for the TREC-8 CLIR task, which may play a role:

	<i>English</i>	<i>French</i>	<i>German</i>	<i>Italian</i>
TREC-7	1689	991	917	501
TREC-8	956	578	717	170

Query Track

JHU/APL also participated in the query track. We generated four query sets, although space aliens in black helicopters managed to prevent two of them from appearing in the official query track collection. These latter two were generated by hand, by reading the narrative version of each source query and generating a title query and a description query for each. Our first official query set (APL5a) was created using a variant of the mutual information statistic [Church & Hanks 1990] to extract important terms from the top 75 documents retrieved for the source query. Our second set (APL5b) used the same statistic to extract important terms from the query track training set. All terms in these query sets were unstemmed words; we did not anticipate that other systems could make use of n-grams.

We used a single system configuration to process each of the 23 query track query sets. This configuration used unstemmed words as terms, and a hidden Markov model to gauge document similarity.

Our results showed tremendous variability in result quality across the 23 query sets. The following table shows HAIRCUT's performance on each query set, the average performance over all eight runs submitted by five different groups to the query track, and HAIRCUT's percentage above or below the all systems average. Our best results were obtained from APL5b, which was developed using training data. For further details on the query track, see the Query Track Overview paper in these proceedings.

	<i>HAIRCUT Average precision</i>	<i>Rel. ret.</i>	<i>Prec. @100</i>	<i>All systs. average prec.</i>	<i>Diff. in avg.</i>
acs1a	0.2521	5553	0.3316	0.2449	2.9%
APL5a	0.2471	5648	0.3300	N/A	N/A
APL5b	0.2912	6242	0.4192	N/A	N/A
INQ1a	0.1697	4261	0.2242	0.1771	-4.2%
INQ1b	0.2089	4902	0.2642	0.2124	-1.6%
INQ1c	0.2261	5195	0.2934	0.2259	0.1%
INQ1d	0.2061	4901	0.2788	0.1919	7.4%
INQ1e	0.2300	4480	0.2796	0.2269	1.4%
INQ2a	0.1622	4201	0.2180	0.1612	0.6%
INQ2b	0.1990	4673	0.2586	0.1869	6.5%
INQ2c	0.2658	5427	0.3386	0.2407	10.4%
INQ2d	0.2208	5367	0.2862	0.2022	9.2%
INQ2e	0.2533	5440	0.3222	0.2388	6.1%
INQ3a	0.1486	3725	0.2050	0.1320	12.6%
INQ3b	0.1292	3766	0.1980	0.1182	9.3%
INQ3c	0.1293	3430	0.1716	0.1299	-0.5%
INQ3d	0.1601	4220	0.2078	0.1578	1.5%
INQ3e	0.1758	4470	0.2382	0.1754	0.2%
pir1a	0.2603	5999	0.3590	0.2464	5.6%
Sab1a	0.2550	5610	0.3312	0.2384	7.0%
Sab1b	0.2791	5946	0.3636	0.2504	11.5%
Sab1c	0.2405	5555	0.3040	0.2533	-5.1%
Sab3a	0.2627	5816	0.3558	0.2364	11.1%

Conclusions

We had good results using a simple linear combination of scores across several HAIRCUT runs. In general, HMMs outperformed the vector model, while n-grams and words were roughly comparable on average. Using run combination, HAIRCUT sees an 8% relative gain on the ad hoc task over the best base run. Such combination is low risk; we have never found a drop in average precision over fifty or more queries. Furthermore, effective combination does not require disparate systems. A single system can produce the required variability simply by using different term types or similarity metrics.

Availability of both words and n-grams also helped us significantly in the cross-language task, for which

HAIRCUT was a first-time participant. HAIRCUT exhibited 79% recall at 1000 on the CLIR task, and a high average precision relative to retrieval using human-translated queries.

References

[Allan *et al.*, 1998] James Allan, Jamie Callan, W. Bruce Croft, Lisa Ballesteros, Don Byrd, Russell Swan, and Jinxi Xu, 'INQUERY does battle with TREC-6.' In E. M. Voorhees and D. K. Harman, eds., *The Sixth Text REtrieval Conference (TREC-6)*. NIST Special Publication 500-240, pp. 169-206, 1998.

[Church & Hanks 1990] Kenneth Ward Church and Patrick Hanks, 'Word association norms, mutual information, and lexicography.' *Computational Linguistics* 16(1):22-29, 1990.

[Mayfield & McNamee 1999] James Mayfield and Paul McNamee, 'Indexing using both n-grams and words.' In E. M. Voorhees and D. K. Harman, eds., *The Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication 500-242, pp. 419-423, 1999.

[Miller *et al.*, 1999] David R. H. Miller, Tim Leek and Richard M. Schwartz, 'BBN at TREC7: Using Hidden Markov Models for Information Retrieval.' In E. M. Voorhees and D. K. Harman, eds., *The Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication 500-242, pp. 133-142, 1999.

[Walker *et al.*, 1998] S. Walker, S. E. Robertson, M. Boughanem, G. J. F. Jones and Karen Sparck Jones, 'Okapi at TREC-6, Automatic ad hoc, VLC, routing, filtering and QSDR.' In E. M. Voorhees and D. K. Harman, eds., *The Sixth Text REtrieval Conference (TREC-6)*. NIST Special Publication 500-240, pp. 125-136, 1998.

Novel Query Expansion Technique using Apriori Algorithm

A. Rungsawang, A. Tangpong, P. Laohawee, T. Khampachua
{fenganr,g4165239,g4165221,g4165248}@ku.ac.th

Massive Information & Knowledge Engineering
Department of Computer Engineering
Faculty of Engineering
Kasetsart University, Bangkok, Thailand

Abstract. One problem in query reformulation process is to find an optimal set of terms to add to the old query. In our TREC experiments this year, we propose to use the association rule discovery (especially apriori algorithm) to find good candidate terms to enhance the query. These candidate terms are automatically derived from collection, added to the original query to build a new one. Experiments conducted on a subset of TREC collections gives quite promising results. We achieve a 19% improvement with old TREC7 adhoc queries.

1 Introduction

Enriching a user's query with synonyms or related terms can improve search performance in a text retrieval system. There are at least two methods to reformulate a search query. The first one is to use relevance feedback where related terms come from the contents of user-identified relevant documents, or pseudo-relevance feedback where expanded terms come from the top k retrieved documents which are assumed to be relevant [4]. The second one is to include terms from an online thesaurus [6] or manually selected terms [3] to the old query. However, these two methods (except the pseudo-relevance feedback) involve with the presence of the user or an additional knowledge source.

We concentrate in our TREC experiments this year with a novel query enhancement technique. Additional terms appended to the original query are obtained from applying apriori algorithm, an association rule discovery used in data mining to extract some useful rules from a large database, to a subset of TREC collections. We have not obtained any promising result with the new adhoc TREC8 query set yet, but achieved 19% improvement with the old adhoc TREC7 queries using our DSIR retrieval system [5].

Our report has been organized in the following way. Section 2 introduces briefly the apriori algorithm, and shows how we apply it to a subset of TREC collections. Section 3 gives detail how we set up our experiments and provides preliminary results. Finally, section 4 concludes this report.

2 Applying Apriori algorithm to TREC Collection

The apriori algorithm, introduced by Agrawal [1], has been widely used to mine useful knowledge in large transaction databases. Typically, a transaction is a list of items (or goods) purchased by a customer during a visit in a store or supermarket. Knowledge are derived in terms of a list of association rules such as “95% of customers who purchase tires and auto accessories also later get automotive services done”, or a formal rule like “85% of customers who buy product A and B also buy product C and D. Discovering all such customer buying patterns is valuable for cross-marketing and attached mailing applications. Other applications include catalog design, product placement, customer segmentation, etc., based on their buying patterns [2].

The problem description of mining association rules can be given as follows. The *support* of a set of items (called later *itemset*) in a transaction database is the fraction of all transactions containing the itemset. An itemset is called *frequent* if its support is greater or equal to a user-specified *support threshold*. An *association rule* is an implication of the form $X \Rightarrow Y$ where X and Y are disjoint itemsets. The support of this rule is the support of $X \cup Y$. The *confidence* of this rule is the fraction of all transactions containing X that also contain Y (i.e. the support of $X \cup Y$ divided by the support of X). From the second example above, the “85%” is the confidence of the rule $\{A,B\} \Rightarrow \{C,D\}$. Given a set of transactions, the problem of mining association rules is to find all association rules that have support and confidence greater than the user-specified minimum support, and minimum confidence respectively [2].

To apply association rule mining to our query reformulation problem, we assume that each document can be seen as a transaction while each separate word inside can also be seen as item or product bought by a customer. Applying apriori algorithm to a TREC collection with specified minimum and maximum support, and confidence will produce a set of association rules in form of $X \Rightarrow Y$, where X and Y are disjoint set of related words (or *wordset*). An enhanced query is then reformulated by adding all words in wordset Y if the wordset X appears in the original query.

3 Preliminary Experiments and Results

We still use DSIR [5] retrieval system in our experiments. Since each pass of association rule discovery takes several hours on a Pentium class machine, we then choose to conduct the experiments using only the FT (Financial Times) collection. FT is first preprocessed in form of transaction database before applying apriori algorithm. All derived rules are employed in the query expansion process. The minimum and maximum support, and confidence are then the additional indexing parameters. Table 1 below gives some results of our experiments.

The first row in Table 1 gives average precision concluded from a set of experiments using original TREC7 queries to search in FT collection with varying document-query weighting combinations. We choose this row as the baseline. The other rows gives results when our new query expansion technique has been

Run	lnc.ntc	lnc.atc	lnc.anc	lnc.ltc
no expansion	0.0722	0.0606	0.0458	0.0835
10-40-10	0.0857 (+18.70%)	0.0658 (+8.58%)	0.0333 (-27.29%)	0.0737 (-11.74%)
10-40-30	0.0861 (+19.25%)	0.0631 (+4.13%)	0.0428 (-6.55%)	0.0725 (-13.17%)
10-40-40	0.0736 (+1.94%)	0.0624 (+2.97%)	0.0425 (-7.21%)	0.0893 (+6.95%)
10-40-50	0.0727 (+0.69%)	0.0610 (+0.66%)	0.0429 (-6.33%)	0.0835 (0%)

Table 1. Query expansion results.

applied. The series of numbers, for example "10-40-10", are the percentage of minimum and maximum support, and the percentage of confidence parameters used in apriori algorithm.

4 Conclusion

Table 1 illustrates very promising results. We can achieve 19% improvement with this query expansion technique, without using any additional knowledge source (i.e. thesaurus, or relevance data) or any user intervention. Since, in our TREC experiments this year, we are quite shortage of time and computing resource, we then unfortunately do not finish our test and has any result on the TREC8 query set.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Database. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207-216, Washington D.C., 1993.
2. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, 1994.
3. P.G. Anick, J.D. Brennan, R.A. Flynn, D.R. Hanssen, B. Alvey, and J.M. Robbins. A Direct Manipulation Interface for Boolean Information Retrieval via Natural Language Query. In Vidick J.L., editor, *Proceedings of the 13th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Bruxelles, Belgium, September 1990.
4. C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic Query Expansion Using SMART: TREC 3. In D. Harman, editor, *The Third Text REtrieval Conference (TREC-3)*. NIST Special Publication 500-225, 1995.
5. A. Rungsawang. DSIR: The First TREC-7 Attempt. In E. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, Gaithersburg, USA, November 1988. National Institute of Standards and Technology, NIST Special publication.
6. E.M. Voorhees. On Expanding Query Vectors with Lexically Related Words. In D. Harman, editor, *The Second Text REtrieval Conference (TREC-2)*. NIST Special Publication 500-215, 1994.

Experiments on the TREC-8 Filtering Track

Keiichiro Hoashi, Kazunori Matsumoto,
Naomi Inoue, and Kazuo Hashimoto
{hoashi,matsu,inoue,kh}@kddlabs.co.jp
KDD R&D Laboratories, Inc.
2-1-15 Ohara Kamifukuoka, Saitama 356-8502, Japan

1 Introduction

For this year's TREC, KDD R&D Laboratories focused on the adaptive filtering experiments of the Filtering Track. The main focus of our research was the development and evaluation of the profile updating algorithm.

Our profile updating algorithm is based on the query expansion method based on *word contribution*[1][2]. Given manual feedback, our QE method has achieved high performance in the ad hoc track. Therefore, our hypothesis is that this method should work well in the filtering track. We will describe how we implemented this method to the filtering track, and analyze experiments.

Our officially submitted results to TREC were generated from a system with a major bug. The results described in this notebook paper are based on the bug-fixed version of our system.

2 Profile updating

As mentioned in Section 1, the query expansion method based on word contribution was implemented for the filtering track. However, some adjustments were necessary for this implementation. In this section, we will explain the basic idea of this method, describe the adjustments made for the filtering track, and present experiment results.

2.1 Definition of word contribution

Word contribution is a measure which expresses the influence of a word (or term) to the similarity between the query and a document. This is defined by the following formula:

$$Cont(w, q, d) = Sim(q, d) - Sim(q'(w), d'(w)) \quad (1)$$

where $Cont(w, q, d)$ is the contribution of the word w in the similarity between query q and document d , $Sim(q, d)$ is the similarity between q and d , $q'(w)$ is query q excluding word w , and $d'(w)$ is document d excluding word w . In other words, the contribution of word w is the difference between the similarity of q and d , and the similarity of q and d when word w is assumed to be nonexistent in both data. Therefore, there are words which have positive contribution, and words which have negative contribution. Words with positive contribution raise similarity, and words with negative contribution lower similarity.

2.2 Analysis of word contribution

Figure 1 illustrates the contribution of all words from a query and a document relevant to it. The data is sorted in descending order according to the contribution of each word.

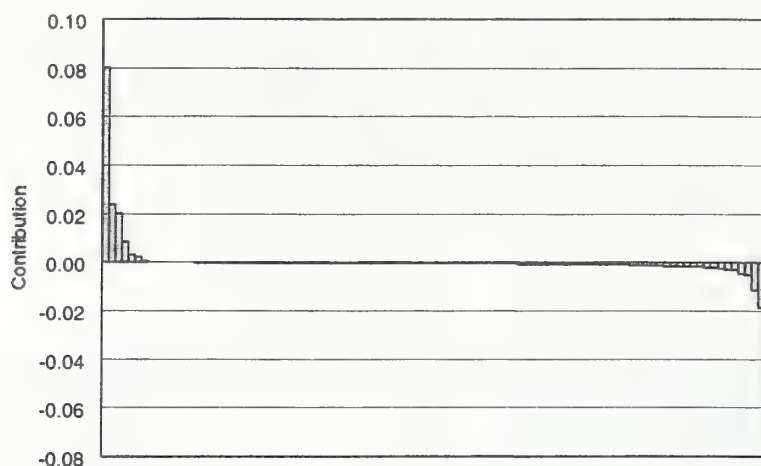


Figure 1: Word contribution between Topic 313 and FBIS3-30043

From Figure 1, it is apparent that there are only a small number of words with highly positive contribution, and a small number of words with highly negative contribution. On the contrary, most words have a contribution near zero, meaning most words do not have a significant influence on the query-document similarity.

As obvious from the definition of word contribution, words with highly positive contribution are presumed to be words that co-occur in the query and document. Such words can be considered as informative words of document relevance to the query. On the contrary, words with highly negative contribution which do not occur in the original query can be considered as words which discriminate relevant documents from other non-relevant documents contained in the data collection.

Since the main objective of query expansion is to add words which are effective in distinguishing relevant documents from the data collection, we assumed that words with highly negative contribution are extremely suitable for expanding the original query. Moreover, we presumed that value of word contribution is a measure of the importance the word has for discrimination. Therefore, the application of word contribution values as the weight of the extracted word for query expansion should also be effective.

2.3 Query expansion method

Based on our arguments in the previous section, we have developed the following query expansion method.

First, the word contribution of all words in the query and relevant documents are calculated. If there are Num documents which are relevant to the query q , the relevant document set for q is $D_{rel}(q) = \{d_1, \dots, d_{Num}\}$. From each relevant document d_i , N words with the lowest contribution are extracted.

Next, a score for each extracted word w is calculated by the following formula:

$$Score(w) = wgt \times \sum_{d \in D_{rel}(q)} Cont(w, q, d) \quad (2)$$

where wgt is a parameter with a negative value (since the contribution is also negative). Calculated scores are regarded as the term frequency values of each word. Therefore, when using the TF*IDF method, the IDF of the word is multiplied to this score to get its final weight. Finally, all extracted words and their weights are added to the original query. If any of the extracted words occur in the original query, that word is not added to the new query. Words with negative scores were also excluded from the expanded query.

2.4 Implementation to profile updating

The query expansion method described in the preceding section requires a “set” of relevant documents. However, since documents come into the system one by one in the filtering process, this set of relevant documents cannot be made unless the system accumulates results. We did not apply this method to our system. Instead, we calculated the word contribution of selected words occurring in retrieved documents, and added them to the profile.

Although our query expansion method proved to be effective without the use of information from non-relevant documents, we felt the necessity to use this information for the filtering process. Therefore, we took a Rocchio-like approach[3] to apply non-relevant document information to the profile. First, the weights of each selected word from non-relevant documents were calculated by the same method as with relevant documents. Next, instead of adding the calculated weight, we subtracted it from the original profile. Words with negative weights resulting from this process are not used for similarity calculation, but all weights are preserved in the profile vector. Therefore, words extracted from both relevant and non-relevant documents have smaller weights than words which are only extracted from relevant documents.

2.5 Additional System Details

Our system is based on the vector space model. The weighting calculation scheme is based on the TF*IDF based weighting formulas for the SMART system at TREC-7 [4], with minor customizations. The TF and IDF factors for our system are as the following:

- TF factor

$$\log(1 + tf) \quad (3)$$

- IDF factor

$$\log \left(\frac{M}{df} \right) \quad (4)$$

where tf is the term's frequency in the document, df is the number of documents that contain the term, and M is the total number of documents in the data collection. The document frequency data was generated from TREC CD-ROMs Vol 4 and 5, excluding (of course) the Financial Times documents. We added 1 to the term frequency inside the logarithm of the TF factor because the tf value resulting from word contribution occasionally has values below 1, which results in a negative weight.

Different weights were set for the calculation of scores from word contribution data, based on the relevance of the document the word was extracted from. Hereafter, w_{rel} expresses the weight for words extracted from retrieved relevant documents, and w_{nrel} expresses the weight for words extracted from retrieved non-relevant documents.

Moreover, we did not make use of the controlled-language field of the Financial Times database for our experiments.

3 Profile updating experiments

In this section, we will describe the experiments made for evaluation of the profile updating algorithm.

3.1 Experiment conditions

The main interest of our experiment is the influence of the 2 parameters of our algorithm, w_{rel} and w_{nrel} , to filtering results. In this experiment, we set w_{rel} to $\{-200, -400, -800\}$, w_{nrel} to $\{-100, -200, -400, -800\}$, and ran all possible parameter combinations (12 runs). For comparison, we also made an experiment run without profile updating.

3.2 Results

Table 1 shows the average scaled utility[5] of each run for all combinations of w_{rel} and w_{nrel} . For comparison, the average scaled utility of the run without profile updating ("*NoPU*") is also shown in this table. The parameter s used for scaled utility calculation is set to 200.

Table 1: Average scaled utility ($s=200$), FT92-94

w_{rel}	w_{nrel}			
	-100	-200	-400	-800
-200	0.4558	0.4840	0.5091	0.5257
-400	0.4172	0.4777	0.5107	0.5184
-800	0.3815	0.4349	0.4842	0.5100
<i>NoPU</i>	0.3807			

As obvious from Table 1, we achieved significant improvement of performance compared to the *NoPU* run, except when the absolute value of w_{rel} is much higher than that of w_{nrel} , as in the case when $\{w_{rel}, w_{nrel}\} = \{-800, -100\}$. Furthermore, scaled utility constantly improves as the absolute value of w_{nrel} increases.

3.3 Discussions

Although we have achieved overall improvement by applying query expansion based on word contribution to our filtering system, the utility is not satisfactory. For further analysis, we examined the results for each year of Financial Times data (FT92, FT93, FT94). Figures 2 - 4 illustrate the improvement of average scaled utility for each year compared to the *NoPU* run.

As observed from the analysis of yearly results, the profile seems to improve if sufficient information from relevant documents are fed back to the profile. However, the excessive retrieval of non-relevant documents before sufficient retrieval of relevant documents lowers the total utility, resulting in the total decline of performance.

The main cause of this problem is that our algorithm does not utilize information from non-relevant documents except for the Rocchio-like approach described in Section 2.4. Subtracting the weights of words extracted from non-relevant documents contributes to higher performance by lowering the influence of words which occur in both relevant and non-relevant documents. However, this does not affect the profile until a relevant document has been retrieved, since negative values in the profile vector are not used for similarity calculation.

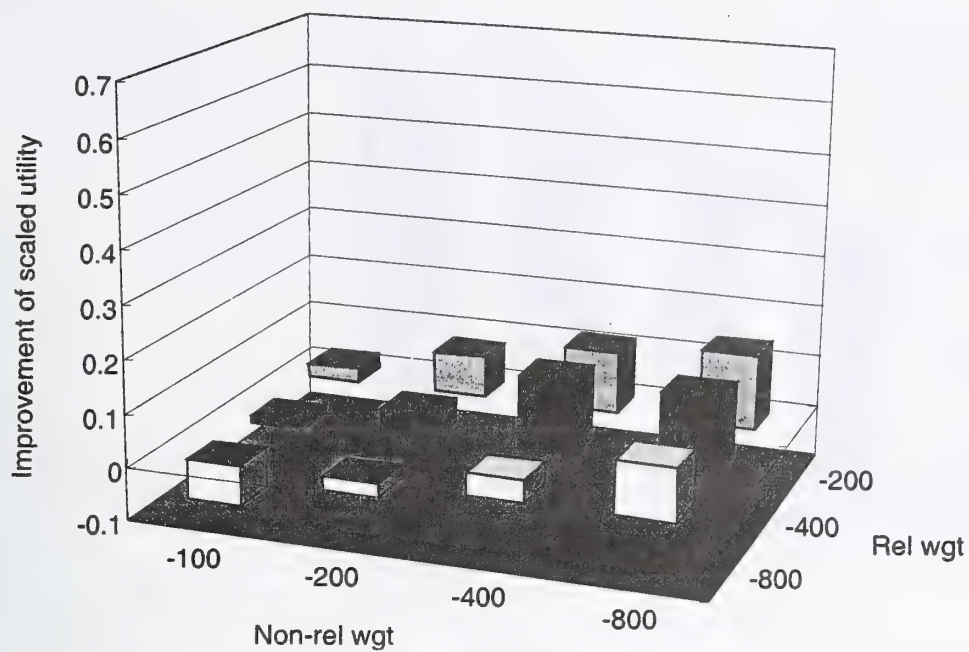


Figure 2: Improvement of scaled utility compared to *NoPU* (FT92)

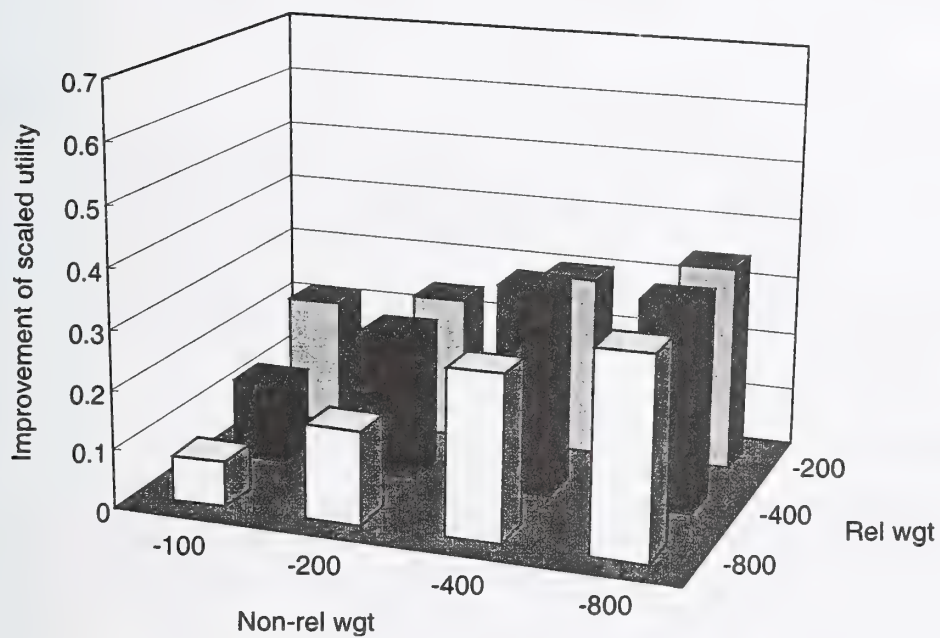


Figure 3: Improvement of scaled utility compared to *NoPU* (FT93)

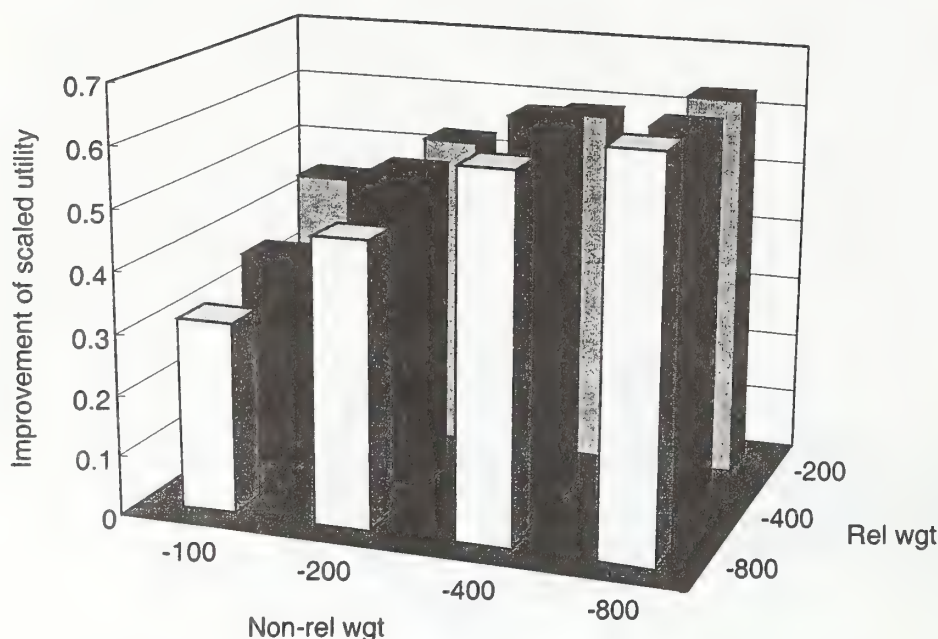


Figure 4: Improvement of scaled utility compared to *NoPU* (FT94)

Therefore, it is necessary to develop a method which makes more use of information from non-relevant documents. Such a method should decrease the number of mistakenly retrieved documents without affecting the retrieval of relevant documents.

4 Conclusion and Future Work

We have made experiments to evaluate the profile updating algorithm and the threshold adjustment algorithm. Experiments on profile updating showed promising results, although there is a necessity to improve our algorithm to apply more information from mistakenly retrieved non-relevant documents to the profile.

We are currently working on an algorithm which makes the use of a profile which expresses the features of past selected non-relevant documents. We hope to evaluate this method in future TREC filtering experiments.

Acknowledgments

The authors appreciate Akiko Onishi of Waseda University and Rickard Johansson of Uppsala University for their great efforts on the experiments described in this paper. We also appreciate the fellow researchers in the Knowledge-Based Information Processing Lab of KDD R&D Laboratories for their advice.

References

- [1] K Hoashi, K Matsumoto, N Inoue, K Hashimoto: "TREC-7 Experiments: Query Expansion Method Based on Word Contribution", The 7th Text REtrieval Conference, NIST SP 500-242, pp 433-441, 1999.
- [2] K Hoashi, K Matsumoto, N Inoue, K Hashimoto: "Query Expansion Method Based on Word Contribution", Proceedings of SIGIR'99, pp 303-304, 1999.
- [3] J Rocchio: "Relevance Feedback in Information Retrieval", in "The SMART Retrieval System - Experiments in Automatic Document Processing", Prentice Hall Inc., pp 313-323, 1971.
- [4] A Singhal, J Choi, D Hindle, D Lewis, and F Pereira: "AT&T at TREC-7", The Seventh Text REtrieval Conference, pp 239-251, 1999.
- [5] D Hull: "The TREC-7 Filtering Track: Description and Analysis", The 7th Text REtrieval Conference, NIST SP 500-242, pp 33-56, 1999.

QALC - the Question-Answering program of the Language and Cognition group at LIMSI-CNRS

Olivier Ferret Brigitte Grau Gabriel Illouz Christian Jacquemin
Nicolas Masson
LIMSI-CNRS
BP 133, 91403 ORSAY Cedex, FRANCE
{ferret,grau,gabrieli,jacquemin,masson}@limsi.fr

1 Introduction

In this report we describe the *QALC system* (the Question-Answering program of the Language and Cognition group at LIMSI-CNRS) which has been involved in the QA-track evaluation at TREC8. The purpose of the Question-Answering track is to find the answers to a set of 200 questions. The answers are text sequences extracted from the volumes 4 and 5 of the TREC collection. All the questions have at least one answer in the collection.

The basic architecture of *QALC* is composed of five parallel modules, two for the questions and three for the corpora, and a sixth pairing module which produces the sentences ranked by decreasing order of relevance. Figure 1 illustrates the main components of *QALC*. More details about each component will be given in the figures corresponding to the detailed descriptions of these subparts.

The *QALC* system relies mainly on genuine Natural Language Processing components. Most of the components rely on a tagged version of the corpus. We use the *TreeTagger* for this purpose (Stein and Schmid, 1995). The system is based on the following six modules:

Natural language question analysis The analysis of the questions relies on a shallow parser which spots discriminant patterns and assigns categories to the questions. The categories correspond to the types of entities which are likely to constitute the answer to this question.

Term extraction The term extractor is based on syntactic patterns which describe compound nouns. The maximal extension of these compounds is produced along with the plausible subphrases.

Automatic indexing & variant conflation Automatic indexing relies on *FASTR* (Jacquemin, 1999), a shallow transformational natural language analyzer which recognizes the occurrences and the variants of the terms produced by the preceding module. Each occurrence or variant constitutes an index to the document which is ultimately used in the process of document ranking and in the process of question/document pairing.

Named entity recognition Similarly, named entities are recognized in the documents in order to build indices which are used for measuring the degree of similarity between the questions and the document sentences. Named entities are extracted through lexico-syntactic patterns combined with significantly large lexical data.

Document ranking & thresholding The last two modules are information retrieval modules as opposed to the four preceding components. Documents are ranked according to a weighted measure of the indices produced by the automatic indexing and variant conflation module. Only the n best ranked documents are selected. A further selection of the documents is made if a plateau can be recognized in the relevance curve of the documents.

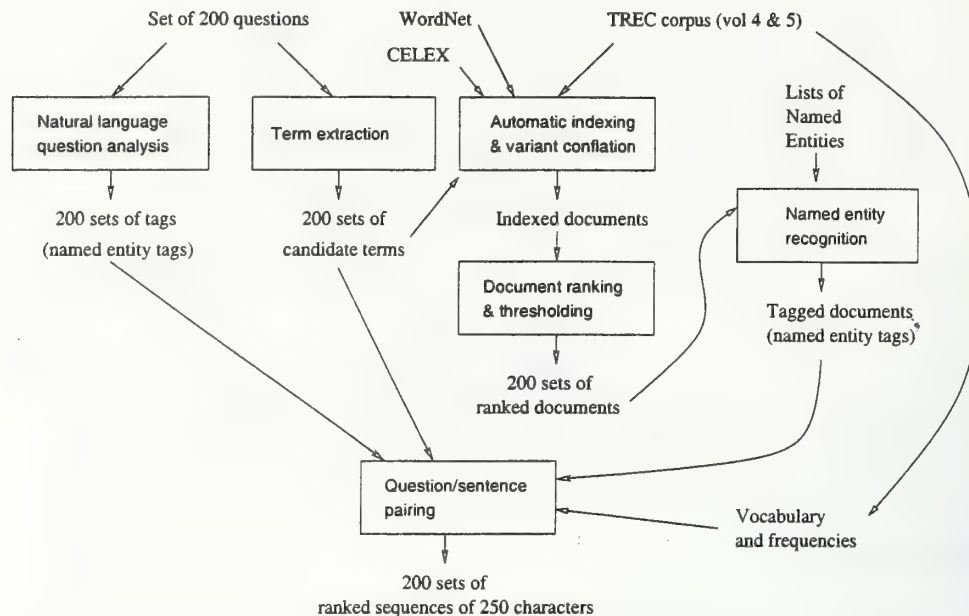


Figure 1: Flowchart of QALC.

Question/sentence pairing Finally, all the data extracted from the questions and the documents by the preceding modules is used by a pairing module to evaluate the degree of similarity between a document sentence and a question. The sentences are chosen from the documents selected by the preceding module.

The following sections present in detail the modules of the *QALC* system.

2 Natural Language Question Analysis

Question analysis is performed in order to assign features to questions and use these features for the pairing measurement between a query (question) and potential answer sentences (answer). Basically, question analysis allows the prediction of the kind(s) of answer, called *target* (for instance, ORGANIZATION). The retrieved documents (see Section 4.2) are processed in order to recognize the named entities. Named entities are labeled with the same set of tags as the questions. During the pairing measurement, the more the question and a sentence share the same tags, the more they are considered as involved in a question-answer relation.

Example:

Question: *How many people live in the Falklands?* → target = NUMBER

Answer: ... *Falklands population of <b_numex_TYPE=NUMBER> 2,100 <e_numex> is concentrated.*

2.1 Target Set

The targets used are PERSON, ORGANIZATION, LOCATION (either CITY or PLACE), TIME-EXPRESSION (either DATE, TIME, AGE or PERIOD), and NUMBER (either LENGTH, VOLUME, DISTANCE, WEIGHT, PHYSICS or FINANCIAL). Some examples of sentences which can be associated with targets follow. Elements of sentences (called *triggers*) which are relevant to assign a given target are underlined.

PERSON: *Who was the first President of the USA?*

ORGANIZATION: *What laboratory discovered the AIDS virus?*

LOCATION:

PLACE: *What is the longest river in Asia? What is the name of the highest mountain in the world?*

TIME-EXPRESSION:

PERIOD: *During which period did the dinosaurs vanish?*

2.2 Types of Questions

Question analysis is performed by a specialized shallow parser. It is based on lexical, syntactic and semantic knowledge, i.e. some specific words, syntactic categories, grammar rules and semantic classes for nouns. We have identified six kinds of formulations for questions allowing the prediction of the kind of answer.

TYPE 1: the answer only depends on the interrogative pronoun. It is the case with *who/whom/whose, where and when*.

The following three patterns are dedicated to parse Which/What-questions. These patterns are identical whatever kind of answer is expected. Disambiguation is provided by the semantic class of the head noun belonging to the noun phrase (NPsem). These classes are detailed in section 2.3.

TYPE 2: *what/which ...be* NPsem

TYPE 3: *what/which* NPsem. A variant of this last rule is TYPE 3b, for questions about time. TYPE 3b: (PREP) *what/which* NPtime

TYPE 4: *what/which is the name of* NPsem.

The last two types entail the analysis of How-questions. TYPE 5 leads the system to choose a target according to the semantic category of the adjective, and, in TYPE 6 questions, the choice relies on the semantic category of the NP.

TYPE 5: *how* AdjSem.

TYPE 6: *how much/many* NPsem.

Our parser first tries to apply these rules on the beginning of the sentence. If none of them is fired, the parser tries to find one of these patterns inside the sentence, as in the sentence *The Faroes are part of what northern European country?* Some rules lead to find several targets in case of ambiguity. For example, the target of the question *Where is Bolivia?* may be either a LOCATION-STATE or a LOCATION-CITY.

2.3 Semantic Categories

Each semantic category corresponds to a target. The category of a noun phrase is based on the semantic class of its main noun. A noun phrase is made of a determiner (DET), followed by successive adjectives (JJ), nouns (N) with a possible possessive case (Poss), or verbs at the gerundive (VBG) or past participle (VBD) form. We have decided to search for the largest noun phrase structure:

$$(\text{DET}) (\text{JJ} | \text{N}_{\text{Poss}} | \text{N} | \text{V}_{\text{BG}} | \text{V}_{\text{BD}})^* \text{N} \quad (1)$$

The head of a noun phrase is the last noun of the longest match. For example, in *Johnny Mathis' high school track coach*, *coach* is recognized as the head. We have established 17 semantic classes, hierarchically structured as shown in Figure 2.

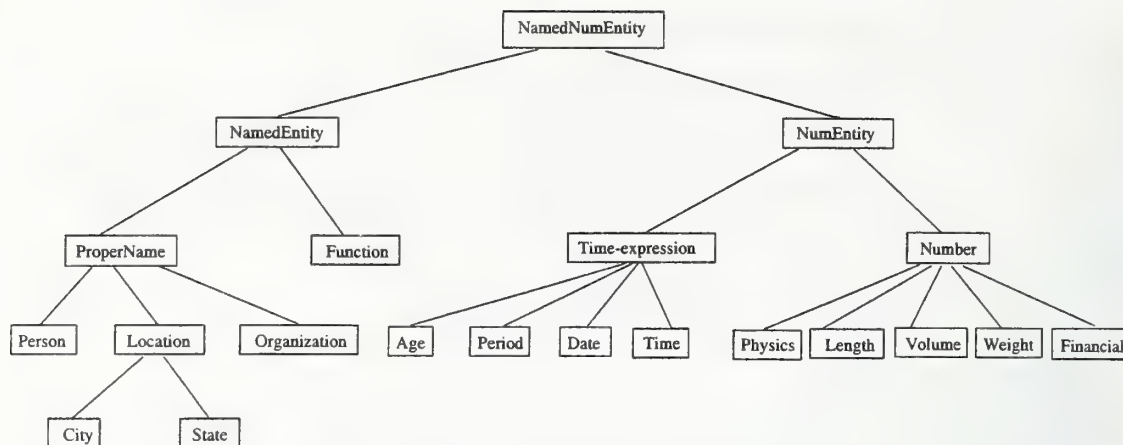


Figure 2: The semantic classes

Let us give some examples to illustrate our recognition of NPs. The NP is in *italics*, its head is underlined and the semantic category of the NP is given after the arrow:

What is *the number* of buffaloes → NUMBER

What *two companies* ... → ORGANIZATION

What *date/year* → DATE

Which *country* → STATE

What is *the world record time* → TIME

What is *the capital* of ... → CITY

Some adjectives used with *how*, as *tall*, *long*, etc., are also classified in semantic classes that belong to the sub-tree representing numeric entities.

3 Term Extraction

The relevant documents are retrieved through a pairing procedure which relies on a measure of similarity. The similarity between a document and a query, detailed in Section 6, is computed on the basis of common terms and on the correspondence between the tags assigned to the question and the named entities extracted from the sentences. Terms are extracted from the questions and sentences are indexed by these terms. In this section, we describe the acquisition of terms from the questions. In the next section, the process of indexing sentences with these terms is detailed.

As for automatic acquisition of terms from questions, we use a simple technique of filtering through patterns of part-of-speech categories. No statistical ranking is possible because of the small size of the questions from which terms are extracted.

The questions are first tagged with the help of the *TreeTagger*. Then, patterns of syntactic categories are used to extract terms from the tagged corpora. They are very close to those in (Justeson and Katz, 1995), but we do not include post-posed prepositional phrases. The pattern used for extracting terms is¹:

$$((((((JJ|N_N|N_P|V_{BG}))?(JJ|N_N|N_P|V_{BG}) (N_P|N_N))))((V_{BD})|((N_N)|(N_P)|(CD))) \quad (2)$$

The longest string is acquired first and substrings can only be acquired if they do not begin at the same word as the superstring. For instance, from the sequence *name_{N_N} of_{IN} the_{DT} US_{N_P} helicopter_{N_N} pilot_{N_N} shot_{V_{BD}} down_{RP}*, the following four terms are acquired: *US helicopter pilot*, *helicopter pilot*, *pilot*, and *shoot*.

The mode of acquisition chosen for terms amounts to considering only the substructures that correspond to an attachment of modifiers to the leftmost constituents (the closest one). For instance, the decomposition

¹ N_N are common nouns, N_P proper nouns and CD numeral determiners.

of *US helicopter pilot* into *helicopter pilot* and *pilot* is equivalent to extracting the subconstituents of the structure $[US [helicopter [pilot]]]$.

4 Automatic Indexing and Document Ranking

The selection of relevant documents relies on an NLP-based indexing composed of both single-word and phrase indices and linguistic links between the occurrences and the original terms. The original terms are those extracted from the questions and presented in Section 3. The tool used for extracting text sequences that correspond to occurrences or variants of these terms is *FASTR* (Jacquemin, 1999). The ranking of the documents relies on a weighted combination of the terms and variants extracted from the documents.

4.1 NLP-based Indexing through *FASTR*

The automatic indexing of documents is performed by *FASTR*, a transformational shallow parser for the recognition of term occurrences and variants. The terms acquired in Section 3 are transformed into grammar rules and the single words building these terms are extracted and linked to their morphological and semantic families.

The *morphological family* of a single word w is the set $M(w)$ of terms in the CELEX database (CELEX, 1998) which have the same root morpheme as w . For instance, the morphological family of the noun *maker* is made of the nouns *maker*, *make* and *remake*, and the verbs *to make* and *to remake*.

The *semantic family* of a single word w is the union $S(w)$ of the synsets of WordNet1.6 (Fellbaum, 1998) to which w belongs. A synset is a set of words which are synonymous for at least one of their meanings. Thus, the semantic family of a word w is the set of the words w' such that w' is considered as a synonym of one of the meanings of w . The semantic family of *maker*, obtained from WordNet1.6, is composed of three nouns: {*maker*, *manufacturer*, *shaper*} and the semantic family of *car* is {*car*, *auto*, *automobile*, *machine*, *motorcar*}.

Variant patterns that rely on morphological and semantic families are generated through metarules. They are used to extract terms and variants from the document sentences in the TREC corpus. The following pattern² extracts the occurrence *making many automobiles* as a variant of the term *car maker*:

$$V_M('maker') RP^? PREP^? (ART (N_N|N_P)^? PREP)^? ART^? (JJ|N_N|N_P|V_{BD}|V_{BG})^{0-3} N_S('car') \quad (3)$$

$V_M('maker')$ is any verb in the morphological family of the noun *maker* and $N_S('car')$ is any noun in the semantic family of *car*.

Relying on the above morphological and semantic families, *auto maker*, *auto parts maker*, *car manufacturer*, *make autos*, and *making many automobiles* are extracted as correct variants of the original term *car maker* through the metarule set used for the QA-track experiment. Unfortunately, some incorrect variants are extracted as well, such as *make those cuts in auto* produced by the preceding metarule.

4.2 Document Ranking

The output of NLP-based indexing is a list of term occurrences composed of a document identifier d , a term identifier—a pair $t(q, i)$ composed of a question number q and a unique index i —, a text sequence, and a variation identifier v (a metarule). For instance, the following index:

$$LA092690-0038 \quad t(131, 1) \quad making \text{ many automobiles} \quad NtoV_{SemArg} \quad (4)$$

means that the occurrence *making many automobiles* from document $d = LA092690-0038$ is obtained as a variant of term $i = 1$ in question $q = 131$ (*car maker*) through the variation given in Section 4.1.

²RP are adverbs, PREP prepositions, ART articles, and V verbs.

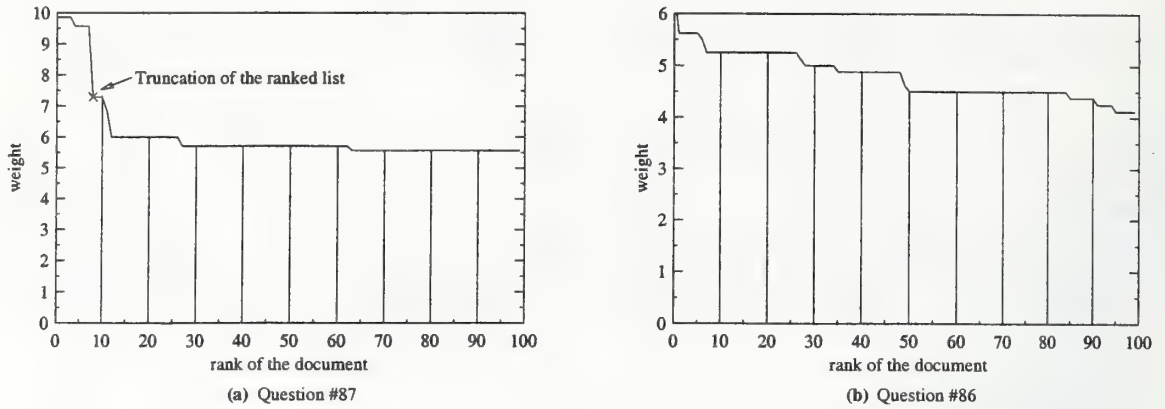


Figure 3: Two types of weighting curve.

Each document d in the collection is associated with a vector $W(d) = (W_q(d))_{q \in \{1, \dots, 200\}}$ of 200 weights, one for each question q . The weighting scheme relies on a measure of quality of the different families of variations described in (Jacquemin, 1999): non-variant occurrences are weighted 3.0, morphological and morpho-syntactic variants are weighted 2.0, and semantic and morpho-syntactico-semantic variants are weighted 1.0.

Since proper names are more reliable indices than common names, each term $t(q, i)$ receives a weight $P(t(q, i))$ between 0 and 1.0 corresponding to its proportion of proper names. For instance, *President Cleveland's wife* is weighted $\frac{2}{3} = 0.66$. Since another factor of reliability is the length of terms, a factor $|t(q, i)|$ in the weighting formula denotes the number of words in term $t(q, i)$.

The weight $W_q(d)$ of a query q in a document d is given by the following formula (5). The products of the weightings of each term extracted by the indexer are summed over the indices $\mathcal{I}(d)$ extracted from document d and normalized according to the number of terms $|\mathcal{T}(q)|$ in query q .

$$W_q(d) = \sum_{(t(q, i), v) \in \mathcal{I}(d)} \frac{w(v) \times (1 + 2P(t(q, i))) \times |t(q, i)|}{|\mathcal{T}(q)|} \quad (5)$$

For each query q , the 100 best ranked documents are retrieved. Mainly two types of weighting curves are observed for the retrieved documents: curves with a plateau and a sharp slope at a given threshold (Figure 3.a) and curves with a slightly decreasing weight (Figure 3.b).

The edge of a plateau is detected by examining simultaneously the relative decrease of the slope with respect to the preceding one, and the relative decrease of the value with respect to the preceding one.

$$\text{if } \frac{W_q(d_2)}{W_q(d_1)} \leq 0.5 \text{ then } i_0 = 2 \quad (6)$$

$$\text{else } i_0 = \min \left\{ i \in \{3, \dots, 100\} \bullet \left(\frac{W_q(d_i) - W_q(d_{i-1})}{W_q(d_{i-1}) - W_q(d_{i-2})} \geq 0.5 \wedge \frac{W_q(d_i)}{W_q(d_{i-1})} \leq 0.8 \right) \right\} \cup \{100\} \quad (7)$$

Through this method, the threshold i_0 is 8 for question #87 (*Who followed Willy Brandt as chancellor of the Federal Republic of Germany?*, Figure 3.a) and 100 for question #86 (*Who won two gold medals in skiing in the Olympic Games in Calgary?*, Figure 3.b). As indicated by Figure 3.a, there is an important difference of weight between documents #8 and #9. The weight of document #8 is 9.57 while the weight of document #9 is 7.29 because the term *Federal Republic* only exists in document #8. This term has a higher weight because it is composed of two proper names.

The i_0 best ranked documents are then processed by the question/sentence pairing module presented in Section 6. At the document level, this module relies on single words, term indices, and named entities. Term indices have been presented in this section, named entities are presented in the next one.

5 Named Entity Recognition

Named entities are recognized in the documents in order to build indices which are used for measuring the degree of similarity between the questions and the document sentences. Named entities receive one of the following types: **PERSON**, **ORGANIZATION**, **LOCATION**, **NUMBER**. They are defined in a similar way to the MUC task (Grishman and Sundheim, 1995) and recognized through a combination of

- lexical lookup (for syntactic or semantic tags on the single words) and rules which use these tags together with lexical elements; and
- dictionary lookup (the direct access to lists of named entities).

The three lists used for lexical lookup are CELEX (CELEX, 1998), a lexicon of 160,595 inflected words with associated lemma and syntactic category, a list of 8,070 first names (6,763 of which are from the CLR (CLR, 1998) archive at New Mexico State University) and a list of 211,587 family names from the CLR archive at New Mexico State University.

5.1 Numeric Entities

This category groups all kinds of number formulations, time expressions, and specialization of numbers according to their units, even if they are not expressed with numbers. Numeric entities are given a tag among **FINANCIAL-AMOUNT**, **LENGTH**, **VOLUME**, **WEIGHT**, **PHYSICS**, **DATE**, **PERIOD**, and **NUMBER**.

The recognition of numeric entities is performed in three steps:

- Firstly, we recognize basic entities as cardinal and ordinal numbers, either written with digits or with letters.
- In a second stage, we apply rules in order to recognize complex numeric entities such as monetary amounts, distances, or weights.
- In a third stage, we apply rules in order to recognize time expressions (labeled as **TIMEX**) such as dates, times, ages and periods. These rules use the output of the first set of rules dedicated to basic numeric entities such as cardinal and ordinal numbers (written in digits or letters).
- In a final stage, all basic numeric entities that have not been included into a complex entity are tagged as **NUMBER**.

5.2 Organizations, Persons and Locations

A list of 22,095 companies from the Wall Street Research Network and 649 organization obtained through lexical acquisition from the Internet is used to spot organization names. In addition, a set of rules recognizes organizations which are noun phrases that either begin with a specific modifier (such as *Christian*, *Democratic*, *Federal*...) or have a specific head word (such as *Academy*, *Administration*, *Association*...).

A word which belongs to the list of proper names exploited during the lexical lookup is tagged as a person. In addition, pairs of capitalized words that begin with a first name, or triples of capitalized words that begin with a title (such as *Dr*, *President*, *Ayatollah*...) are tagged as persons.

Simple anaphora which correspond to the elision of a subpart of a proper name are handled by the recognizer: a rule states that all the occurrences of a $\langle name \rangle$ inside a single document are references to the person identified by the rule $\langle firstName \rangle \langle name \rangle$.

The location recognition module uses two lexical resources from the CLR: a list of 7,813 city names and a list of 1,144 country names. No rules have been written for these entities so far.

6 Question/Sentence Pairing

This section presents the module that selects for each question a list of 5 ranked responses that are no longer than 250 characters each. This module relies on the results of all the preceding modules:

- each question is assigned a set of terms and one or several categories according to its focus;
- a set of documents is selected for each question. In each of them, named entities and terms extracted from the questions are tagged.

6.1 General Principles

The *Question/Sentence Pairing* module relies on two main choices, close to those that support the preliminary version of the Deep Read system (Hirschman et al., 1999). Firstly, we take the sentence unit as our basic unit for the answers to the questions, given that the *QALC* system aims at finding precise information that can be expressed in a single clause. Secondly, the *Question/Sentence Pairing* module directly searches for the possible answers of a question in all the documents selected by the *Document Ranking & Thresholding* module without trying to delimit smaller units. Such a choice allows the scanning of a large set of possible answers without too high a cost.

Considering the two preceding choices, the *Question/Sentence Pairing* module is based on a very simple principle: we compare each sentence from the selected documents for a question to this question and we always keep the five sentences that are the most similar to the question. This comparison is done first by transforming both the question and each document sentence into vectors and then, by computing a similarity measure between these two vectors. As is usual in the field of Information Retrieval, this similarity measure basically relies on the words of both the question and the sentences from the documents. But in our case, it also takes into account and merges all the linguistic information from the preceding modules. The final step of the module consists of cutting down the sentences that are longer than 250 characters. This is done here by simply removing the first and the last characters until we reach the fixed length.

6.2 Basic Similarity Measure

The similarity evaluation starts by turning sentences (questions and document sentences) into vectors of words. Such a vector only contains the most significant words of the primary sentence, i.e. mainly its content words: nouns and proper nouns, verbs, adjectives (including comparatives and superlatives), adverbs (including superlatives and comparatives), foreign words, symbols and cardinal numbers. These words are lemmatized and we take their canonical form as a reference in the vectors. We also have a short stop-list in order to remove some frequent words that are selected according to the previous criterion but that are not meaningful in our task. All this selection process relies on the morpho-syntactic tagging done by the *TreeTagger*, which also performs the lemmatizing.

Each word in a vector is weighted according to its importance in relation to the *Question/Answering* corpus. This importance is evaluated by using the *tf.idf* weighting policy, as it is often done in Information Retrieval. Word order in vectors is not significant because our similarity measure does not take this parameter into account. We think that it is too restrictive a constraint considering the kind of processing we do.

The similarity measure between a vector representing a question, V_q , and a vector representing a document sentence, V_d , is given by:

$$\text{sim}(V_q, V_d) = \frac{\sum_i w d_i}{\sum_j w q_j} \quad (8)$$

with $w q_j$, the weight of a word in the question vector and $w d_i$, the weight of a word in a sentence vector that is also in the question vector.

This measure evaluates the proportion and the importance of the words in the question vector that are present in the sentence vector with regard to all the words in the question vector. It is not symmetrical:

it favors the question point of view. It focuses on the similarities between the questions and the document sentences because firstly, there are too many differences at this level in comparison with the similarities, and then, these differences are globally not relevant.

On the other hand, we take into account the difference in length between a question and a document sentence. This criterion is used as a secondary key for sorting the sentences that are selected as possible answers to a question : if two sentences have the same similarity value, we sort as first the sentence that has the closest length with regard to the question.

6.3 Similarity Measure with Linguistic Features

We have chosen to take into account the results of the previous modules without changing our basic mechanism. Two kinds of linguistic features are considered: terms and named entities. Both of them are added to the vectors as if they were new significant words.

Terms Each of the terms extracted from a question has a unique identifier, which is used for marking the occurrences and the variants of this term both in the questions and in the document sentences. In the sentences, this identifier is associated with a score that reflects the distance between the found variant and its reference form in the question (see Section 4.2). In a question vector, we add the term identifier with a default weight of 0.5. In a sentence vector, we add the identifier of the recognized terms with a weight equal to the score of the variant divided by the maximum possible score.

Named entities Each recognized named entity is marked with a specific tag according to its type (see Section 5). On the other hand, the kind of the answer expected for each question is determined by the *Question Analysis* module. Thus, for a question, we add the tag(s) of the expected type(s) of the answer to its vector and for a sentence, we add the tags of the named entities that have been recognized in the sentence to its vector. In both cases, each tag is given a fixed weight, which is set to 0.5.

The weights associated with the term identifiers and with the named entity tags have been experimentally set. They are globally lower than the weights of the words. Thus, as for the difference of length, these linguistic features are used as a secondary criterion in the similarity measure. They help in increasing the rank of an answer which has already been selected, but the selection of the possible answers is mainly based on the number and the importance of the words shared by the question and these answers.

7 Results

Our official results are the following: our mean reciprocal rank is 0.341, with 88 questions answered. More precisely, we got 56 answers at rank 1, 12 at rank 2, 9 at rank 3, 7 at rank 4 and 4 at rank 5. Globally, this means that when an answer is found, it often appears on top position. Since our sentence truncating procedure for producing 250 character answers is very rough, our system sometimes missed the correct answer although it found a correct sentence. We think that applying very simple heuristics based on the recognized named entities could easily solve this problem.

One important aspect of the Question/Answering task is the ability to determine the expected type of answer (i.e. the target) in order to match it with named entities that are recognized in documents. Hence, we have analyzed our results according to this criterion. Among the 198 questions, a target was identified for 162 of them. The following table shows the percentage of correct answers in relation to the target type.

Target name	Correct answers (%)	Target name	Correct answers (%)
none	47	location	42
time-expression	27	number	65
person	49		

These results lead us to conclude that the recognition of a target does not influence the system's ability to find a correct answer, which seems rather contradictory with the main trends of the best participants. However our results for numbers, that are easily recognizable, comply with these trends and suggest that we have to enhance our named entity recognition module. We also think that our pairing module takes into account too weakly the concordance between the targets of questions and the named entities of documents.

8 Conclusion and Future Developments

Since the LIMSI laboratory did not have experience in the development of Question-Answering system before participating in the QA-track this year, many research issues have arisen from the construction of our system. Among the future developments that we are considering for our next participation in the QA-track are:

- exploring different weighting schemes for the computation of the similarity of a question and a sentence. For instance, when pairing a sentence with a question requiring a date, named entities denoting a date in sentences should be weighted higher;
- answer unit could be enlarged and position of indices inside a document could be accounted for in order to focus on the units that gather the largest number of indices and which are more likely to provide the answer;
- cascaded indexing with *FASTR* would optimize the computation by running *FASTR* on a smaller part of the collection;
- term acquisition could be improved through a disambiguation of long noun phrases and a better part-of-speech tagging of the questions;
- named entity recognition could be improved through machine learning techniques (Baluja, Mittal, and Sukthankar, 1999).

References

- Baluja, Shumeet, Vibhu O. Mittal, and Rahul Sukthankar. 1999. Applying machine learning for high performance named-entity extraction. In *Proceedings, PACLING'99*, pages 365–378, Waterloo, CA.
- CELEX. 1998. www ldc.upenn.edu/readme_files/celex.readme.html. Consortium for Lexical Resources, UPenn.
- CLR. 1998. <http://crl.nmsu.edu/cgi-bin/Tools/CLR/clrcat#D3>. Consortium for Lexical Resources, NMSU, New Mexico.
- Fellbaum, Christiane, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.
- Grishman, R. and B. Sundheim. 1995. Design of the muc-6 evaluation. In NIST, editor, *the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD. NIST, Morgan-Kaufmann Publisher.
- Hirschman, L., M. Light, E. Breck, and J. D. Burger. 1999. Deep read: A reading comprehension system. In *Proceedings, ACL'99*, pages 325–332, University of Maryland.
- Jacquemin, Christian. 1999. Syntagmatic and paradigmatic representations of term variation. In *Proceedings, ACL'99*, pages 341–348, University of Maryland.
- Justeson, John S. and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27.
- Stein, Achim and Helmut Schmid. 1995. Etiquetage morphologique de textes français avec un arbre de décision. *t.a.l.*, 36(1-2):23–36.

The LIMSI SDR System for TREC-8

Jean-Luc Gauvain, Yannick de Kercadio, Lori Lamel and Gilles Adda

Spoken Language Processing Group
LIMSI-CNRS, BP 133
91403 Orsay cedex, FRANCE
{gauvain,kercadio,lamel,gadda}@limsi.fr

ABSTRACT

In this paper we report on our TREC-8 SDR system, which combines an adapted version of the LIMSI 1998 Hub-4E transcription system for speech recognition with an IR system based on the Okapi term weighting function. Experimental results are given in terms of word error rate and average precision for both the SDR'98 and SDR'99 data sets. In addition to the Okapi approach, we also investigated a Markovian approach, which although not used in the TREC-8 evaluation, yields comparable results. The evaluation system obtained an average precision of 0.5411 on the reference transcriptions and of 0.5072 on the automatic transcriptions. The word error rate measured on a 10 hour subset is of 21.5%.

INTRODUCTION

There expansion of different media sources for information dissemination (radio, television, internet) has led to a need for automatic processing tools. Today's methods for audio segmentation, transcription and indexing are manual, with humans reading, listening and watching, annotating topics and selecting items of interest for the user. Even partial automation of some of these activities can allow more information sources to be processed and significantly reduce processing costs while eliminating tedious work. Some application areas that could benefit from automated transcription and indexing technology include the creation and access to digital multimedia libraries (disclosure of the information content and content-based indexing, such as are under exploration in the OLIVE [13] project), media monitoring services (selective dissemination of information based on automatic detection of topics of interest) as well as new emerging applications such as News on Demand (such as the Informedia [10] project) and Internet watch services. Such applications are feasible due to the large technological progress made over the last decade, benefiting from advances in micro-electronics which have facilitated the implementation of more complex models and algorithms.

Automatic speech recognition is a key technology for audio and video indexing, for data such as radio and television

broadcasts. Most of the linguistic information is encoded in the audio channel of video data, which once transcribed can be accessed using text-based tools. This is in contrast to the image data for which no common description language is available.

In this paper we describe the LIMSI spoken document indexing and retrieval system developed for the TREC-8 SDR evaluation. This system combines a state-of-the-art speech recognizer [9] with an Okapi-based IR system. A Markovian-based IR system has also been developed and contrastive experimental results using this system are provided. All of our development work was carried out using the TREC-7 SDR data set (100 hours) and the associated set of 23 queries. This year's SDR task was quite more challenging than the SDR'98 track in that the audio data was increased to about 550 hours of broadcasts, which has strong implications on the transcription process. The next section describes the LIMSI speech transcription system and the modifications made for use in this evaluation, trying to find the best compromise between accuracy and speed. In the following section the two IR systems are presented, and experimental results for various configurations are provided.

TRANSCRIBING BROADCAST NEWS

At LIMSI we have been working on using statistical models to transcribe broadcast news data since 1996. Due to the availability of large audio and textual corpora via the Linguistic Data Consortium (LDC)¹, most of our work on broadcast news transcription has been carried out on American English. In the context of the EC LE OLIVE project [13], broadcast news transcription systems for French and German have recently been developed.

Radio and television broadcast shows are challenging to transcribe as they contain signal segments of various acoustic and linguistic natures. The signal may be of studio quality or may have been transmitted over a telephone or other noisy channel (i.e., corrupted by additive noise and nonlinear distortions), or can contain speech over music or pure

¹<http://www ldc.upenn.edu>

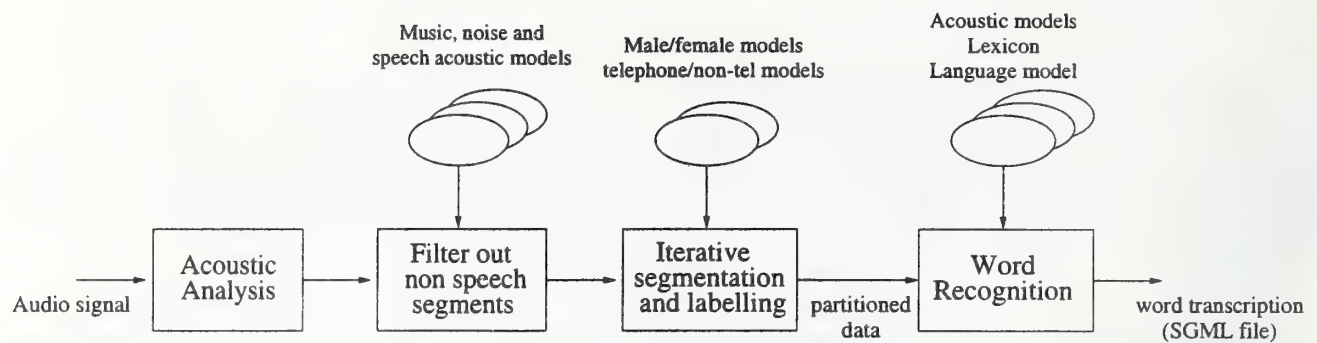


Figure 1: Overview of transcription system for audio stream.

music segments. Gradual transitions between segments occur when there is background music or noise with changing volume, and abrupt changes are common when there is switching between speakers in different locations. The speech is produced by a wide variety of speakers: news anchors and talk show hosts, reporters in remote locations, interviews with politicians and common people, unknown speakers, new dialects, non-native speakers, etc. Speech from the same speaker may occur in different parts of the broadcast, and with different background noise conditions. The linguistic style ranges from prepared speech to spontaneous speech. Acoustic and language modeling must accurately account for this varied data.

Two principle types of problems are encountered in automatically transcribing broadcast news data: those relating to the varied acoustic properties of the signal, and those related to the linguistic properties of the speech. Problems associated with the acoustic signal properties are handled using appropriate signal analyses, by classifying the signal according to segment type and by training acoustic models for the different acoustic conditions. Noise compensation is also needed in order to achieve acceptable performance levels. Most broadcast news transcription systems make use of unsupervised acoustic model adaptation as opposed to noise cancelation, which allow adaptation without an explicit noise model. In order to address the variability observed in the linguistic properties, the differences in speaking styles need to be analyzed with regard to lexical items, word and word sequence pronunciations, and frequencies and distribution of hesitations, filler words, and respiration noises. Once such an analysis is carried out, the variability needs to be accounted for in the acoustic and language models [3].

System overview

The LIMSI SDR'99 transcription system shown in Figure 1, is based on the LIMSI 1998 Hub-4E system which achieved an official word error of 13.6% in the Nov'98 ARPA evaluation. Prior to recognition the audio stream is first partitioned. Data partitioning serves to divide the continuous stream of acoustic data into homogenous segments,

associating appropriate labels with each segment. The segmentation and labeling procedure [4] first detects and rejects non-speech segments, and then applies an iterative maximum likelihood segmentation/clustering procedure to the speech segments. The result of the partitioning process is a set of speech segments with cluster, gender and telephone-band/wideband labels. The speech recognizer uses continuous density HMMs with Gaussian mixture observation densities for acoustic modeling and 4-gram statistics for language modeling. The states of the context-dependent phone models are tied by means of a decision tree.

Audio Partitioner

The goal of partitioning is to divide the continuous audio stream into homogeneous acoustic segments, to remove non-speech segments and to assign bandwidth and gender labels to each segment. The audio partitioning procedure, introduced for the Nov'97 evaluation [4, 5] and used in the LIMSI Nov'98 Hub-4E system [9], is as follows:

1. First, the non-speech segments are detected (and rejected) using Gaussian mixture models (GMMs). Four GMMs, each with 64 Gaussians serve to detect speech, pure-music and other (background). All test segments labeled as music or silence are removed prior to further processing.
2. An iterative maximum likelihood segmentation/clustering procedure is then applied to the speech segments using GMMs and an agglomerative clustering algorithm. Given the sequence of cepstral vectors, the algorithm tries to maximize an objective function which is a penalized log-likelihood. Alternate Viterbi reestimation and agglomerative clustering yields a sequence of estimates with non decreasing values of the objective function. The algorithm stops when no further merges are possible. The cluster size is constrained to ensure that each cluster corresponds to at least 10s of speech. This procedure is controlled by 3 parameters: the minimum cluster size (10s), the maximum log-likelihood loss for a merge, and the segment boundary penalty.

When no more merges are possible, the segment boundaries are refined (within a 1s interval) using the last set of GMMs and an additional relative energy-based boundary penalty. This is done to locate the segment boundaries at silence portions, so as to avoid cutting words.

3. Speaker-independent GMMs corresponding to wideband and telephone speech (each with 64 Gaussians) are then used to label the segment bandwidths. This is followed by segment-based gender identification, using 2 sets of GMMs with 64 Gaussians (one for each bandwidth). The result of the partitioning process is a set of speech segments with cluster, gender and telephone/wideband labels.

Speech Recognizer

As usual [3, 6, 4, 9], the acoustic feature vector contains 39 cepstral parameters derived from a Mel frequency spectrum estimated on the 0-8kHz band (or 0-3.5kHz band for telephone data) every 10ms. For each 30ms frame the Mel scale power spectrum is computed, and the cubic root taken followed by an inverse Fourier transform. Then LPC-based cepstrum coefficients are computed. The cepstral coefficients are normalized on a segment-cluster basis using cepstral mean removal and variance normalisation. Thus each cepstral coefficient for each cluster has a zero mean and unity variance. The 39-component acoustic feature vector consists of 12 cepstrum coefficients and the log energy, along with the first and second order derivatives. Each phone model is a tied-state left-to-right CD-HMM with Gaussian mixtures. The triphone-based context-dependent phone models are word-independent but position-dependent. The tied states are obtained by means of a decision tree.

The decoding procedure of the LIMSI Nov'98 Hub-4E system has been changed in order to reduce the computation time required to process the 550 hours of BN data for the SDR'99 evaluation. Word recognition is performed in three passes:

1. *Word graph generation*: An initial hypothesis and word graph are generated using a small bigram-backoff language model and gender-specific sets of position-dependent cross-word triphones.
2. *3-gram decoding with acoustic model adaptation*: Unsupervised acoustic model adaptation is performed for each segment cluster using the MLLR technique [14] using the initial hypotheses. Each segment is decoded with a trigram language model, the adapted acoustic models, and the word graph.
3. *4-gram decoding with acoustic model adaptation*: The final hypotheses are generated using a 4-gram language

model with acoustic model adaptation using the hypotheses of pass 2.

Acoustic model training

We used the acoustic models of the LIMSI Nov'98 Hub-4E system. These models were trained on about 150 hours of broadcast data (only the official Hub-4E training data from 1995, 1996, and 1997). The acoustic models are position-dependent triphones with about 11500 tied states (366K Gaussians), obtained using a divisive decision tree based clustering algorithm. Two sets of gender-dependent acoustic models were built using MAP [8] adaptation of SI seed models for each of wideband and telephone band speech. A portion of the Hub-4E training data was also used to build the Gaussian mixture models for partitioning (speech, music and noise models) and for gender and bandwidth identification. About 2 hours of pure music portions taken from the acoustic training data were used to estimate the music GMM.

Language model training

The language models of the LIMSI Nov'98 Hub-4E system were used. The language models are fixed and were obtained by interpolation of backoff n -gram language models trained on different data sets. To build the n -gram LM, four models trained on the following sources were interpolated:

1. BN transcriptions from LDC (years 92-95) and from PSmedia (years 96 and 97 (the period 15/10/96 - 14/11/96 was excluded): 203 M words
2. NAB newspaper texts and AP Wordstream texts prior to September 1995: 202 M words
3. NAB newspaper texts and AP Wordstream texts from July 1996 to August 1997 (the period 15/10/96 - 14/11/96 was excluded) : 141 M words
4. Transcriptions of the acoustic data, BN data (including the 1995 MarketPlace data): 1.6M words

The interpolation coefficients of these four LMs were chosen so as to minimize the perplexity on the Nov'96 and Nov'97 evaluation test sets. A backoff 4-gram LM is then derived from this interpolation by merging the four component LMs [20]. Bigram and trigram LMs were built in a similar manner for use in the first two decoding steps.

All words occurring a minimum of 15 times in the broadcast news texts (63,954 words) or at least twice in the acoustic training data (23,234 mots) were included in the recognition vocabulary, resulting in a 65,122 word list. The lexical coverage is 99.5% on the Hub-4E Nov'97 eval test set and 99.1% on the Hub-4E Nov'96 eval test set.

The BN texts from PSmedia (also used for query expansion in our IR system) were processed using a modified version of a perl script from BBN made available by LDC. The

BN training texts were cleaned in order to be homogeneous with the previous texts. These texts were processed so as to treat some frequent word sequences as compound words, and to treat the most frequent acronyms in the training texts as whole words instead of as sequences of independent letters.

Lexicon

Pronunciations are based on a 48 phone set (3 of them are used for silence, filler words, and breath noises). A pronunciation graph is associated with each word so as to allow for alternate pronunciations, including optional phones. The 65k vocabulary contains 65,122 words including 72,734 phone transcriptions. Frequent inflected forms have been verified to provide more systematic pronunciations. As done in the past, compound words for about 300 frequent word sequences subject to reduced pronunciations were included in the lexicon as well as the representation of frequent acronyms as words.

Transcription results

Table 1 reports the word recognition results on the eval test sets from the last three years. All of our system development was carried out using the Hub-4E eval96 and the SDR'98 data set. For the SDR'98 data set we built a system respecting the rules from last year's SDR evaluation. Since the SDR'98 test data is part of the standard Hub-4E training data, acoustic models were trained on only about 80 hours of acoustic data as opposed to 150h. Similarly language models were trained using only those texts predating the test epoch (Jan'98).

The word transcription error is seen to be on the order of 20% on the broadcast data. The better results for the Hub-4E Nov'97 (h4-97) and Nov'98 (h4-98) test sets are due to prior selection of the test data to include a higher proportion of prepared speech. The word error of the SDR'99 is about 15% higher than the LIMSI Nov'98 Hub-4E system. The difference in performance of the SDR'98 and SDR'99 systems can be attributed to the difference in training data.

	Test set (Word Error)				
	h4-96	h4-97	h4-98	sdr98	sdr99
System	1.8 h	3 h	3 h	100 h	10 h
Hub4'98	19.8	13.9	13.6	-	-
SDR	22.6	16.5	16.0	24.4*	21.5

Table 1: Summary of BN transcription word error rates on the 3 last DARPA evaluation test sets (h4-96, h4-97, h4-98) and the SDR'98 and '99 test sets using the LIMSI HUB4'98 system and the LIMSI SDR'99 system (about 15xRT). *Results on the SDR'98 test set were obtained with a system trained on about half the amount of acoustic data and less LM texts, in accordance with the SDR'98 evaluation condition.

INFORMATION RETRIEVAL

Our SDR'99 IR system has been designed following the Okapi approach [18]. In order for the same IR system to be applied to different text data types (automatic transcriptions, closed captions, additional texts from newspapers or newswires), all of the documents are preprocessed in a homogeneous manner. This preprocessing or tokenization, described below, is the same as what is done to prepare text sources for training the speech recognizer language models [7], and attempts to transform them to be closer to the observed American speaking style. There is no stop list, that is to say no words are discarded during the pre-processing stage. The index terms are obtained after translation using a lexicon of stems. Query expansion is obtained via Blind Relevance Feedback (BRF) using both the SDR'99 audio data collection and a parallel text corpus of broadcast news transcripts.

All development was carried out using exclusively the SDR'98 evaluation data, consisting of about 2800 documents with the associated 23 queries. Two approaches for IR were explored, the first based on the Okapi term weighting function and the second using a Markovian one [11, 15]. Due to the limited amount of development data and our limited experience with IR systems, we chose to submit the Okapi-based system for the evaluation even though comparable results were being obtained with the Markovian approach. Some comparative results for the two approaches are given at the end of this section.

The parameter values were chosen to simultaneously optimize performance on automatic recognizer transcripts and the provided manual reference transcriptions. Better IR performance can be obtained if the parameters for the two transcription types are optimized independently, but this would result in two different IR systems. It is also worth noting that the reference transcripts of the SDR'98 data are detailed manual transcriptions, whereas for the SDR'99 data these are closed captions. The different transcript types made us uncertain as to the reliability of our development work.

Tokenization

The tokenizer transforms the texts to a unified format. The basic operations include translating numbers and sums into words, removing all the punctuation signs, removing case distinctions and detecting acronyms and spelled names such as *K.G.B.* However removing all punctuation markers implies that certain hyphenated words such as *anti-communist*, *non-profit* are rewritten as *anti communist* and *non profit*. While this offers advantages for speech recognition, it can lead to IR errors. To avoid IR problems due to this transformation, the output of the tokenizer (and recognizer) is checked for common prefixes, in order to rewrite the sequence of words *anti communist* as a single word. The prefixes that are handled include *anti*, *co*, *bi*, *counter*. A rewrite

lexicon containing compound words formed with these prefixes and a limited number of named entities (such as *Los-Angeles*, *Saint-Tropez*) is used to transform the texts. Similarly all numbers less than one hundred are treated as a single entity (such as *twenty-seven*).

Stemming

In order to reduce the number of lexical items for a given word sense, each word is translated into its stem (as defined in [2, 16]) or, more generally, into a form that is chosen as being representative of its semantic family. The stemming lexicon (using the UMass 'porterized' lexicon) [2] contains about 32000 entries and was constructed using Porter's algorithm on the most frequent words in the collection, and then manually corrected.

The IR term list was limited to 45k entries (after stemming) for implementation reasons. For the SDR'99 audio data collection, this filtering only affected the R1 condition where the least frequent terms were removed.

Baseline search

The score of a document d for a query is given by the Okapi-BM25 formula[17]. It is the sum over all the terms t in the query of the following weights:

$$cw_{t,d} = qtf_t \frac{(K+1) \times tf_{t,d}}{K \times (1-b + b \times L_d) + tf_{t,d}} \log \frac{N}{N_t} \quad (1)$$

where $tf_{t,d}$ is the number of occurrences of term t in document d (i.e. term frequency in document), N_t is the number of documents containing term t at least once, N is the total number of documents in the collection, L_d is the length of document d divided by the average length of the documents in the collection, and qtf_t the number of occurrences of term t in the query.

The parameter values of the Okapi formula were chosen in an attempt to maximize the average precision on the SDR'98 data set. The resulting values were thus a compromise between the optimal configuration for the R1 and S1 conditions, in order to be able to use the same values for both conditions. The S1 transcripts were obtained with a speech recognizer trained on 75 hours of acoustic data and language model training texts predating the test period. The recognition word error rate on this data (using the NIST SDR'98 scoring procedure) was 24.4% (cf. Table 1). The parameters were fixed for all the evaluation conditions at: $b=0.86$; and $K=1.2$ for the baseline run without query expansion, and $K=1.1$ with query expansion.

Query expansion

The text of the query may or may not include the index terms associated with relevant documents. One way to cope

with this problem is to use query expansion based on terms present in retrieved documents on the same (Blind Relevance Feedback) or other (Parallel Blind Relevance Feedback) data collections [19]. We have experimented with both approaches, and our submitted system incorporated both BRF and PBRF using 6 months of commercially available broadcast news transcripts for the period of June-December 1997 [1]. This corpus contains 50 000 stories and 49.5 M words.

For a given query, the terms found in the top B documents from the baseline search are ranked by their offer weight (ow_t), and the top T terms are added to the query. As proposed in [18] the following formula for ow_t was used:

$$ow_t = r_t \log \frac{(r_t + 0.5)(N - N_t - B + r_t + 0.5)}{(N_t - r_t + 0.5)(B - r_t + 0.5)} \quad (2)$$

where r_t is the number of documents (among the B documents) containing the term t .

Since only the T terms with best offer weights are kept, we filtered the terms using a stop list of 144 common words, in order to increase the likelihood that these terms are relevant.

data	base	brf	pbrf	brf+pbrf
R1	0.4689	0.5597	0.5609	0.5803
S1	0.4594	0.5329	0.5442	0.5636

Table 2: Development IR results on the SDR'98 data set ($b=0.86$, $K=1.1$, $B=15$, $T=5$) for the baseline system, and with 3 configurations for query expansion.

Four experimental configurations are reported in Table 2 for the SDR'98 development data: baseline search (*base*), query expansion using BRF (*brf*), query expansion with parallel BRF (*pbrf*) and query expansion using both BRF and PBRF (*brf+pbrf*). For BRF and PBRF, the terms are added to the query with a weight of 1. For BRF+PBRF, the terms from each source are added with a weight of 0.5. The parameter values used for these experiments are the result of our development work. We felt that it was safest to add only a few terms, assuming that only a small number of documents were relevant. Therefore the development experiments compared performance for relatively small values of B and T , with the best performance being obtained with $B = 15$ and $T = 5$. The results reported in Table 2 clearly demonstrate the interest of using both BRF and PBRF expansion techniques with consistent and comparable improvements over the baseline for the two conditions (R1 and S1). As has been previously reported by other sites, there is only a slight performance degradation in going from the R1 condition to the S1 condition, even with a transcription word error of 24%.

Evaluation Results

The parameter setting optimized on the SDR'98 data set (cf. Table 2) were used for all our submissions on the SDR'99 data set. Table 3 summarizes the results of the LIMSIR system for the R1, S1, and cross-recognizer conditions. In addition to the official numbers obtained with query expansion using both BRF and PBRF, the results for the 3 other configurations (no query expansion, query expansion with BRF and query expansion with PBRF) are also provided.

<i>data</i>	<i>base</i>	<i>brf</i>	<i>pbrf</i>	<i>brf+pbrf</i>
R1	0.4711	0.5330	0.5126	0.5411
S1	0.4327	0.4978	0.4848	0.5072
B1	0.4180	0.4787	0.4702	0.4828
B2	0.4212	0.4786	0.4748	0.4839
HTK	0.4436	0.5163	0.4933	0.5176
ATT	0.4178	0.4956	0.4621	0.4925
SHEF	0.4041	0.4659	0.4593	0.4787
CMU	0.2732	0.2980	0.3368	0.3234

Table 3: LIMSIR official IR results on the SDR'99 data set ($b=0.86$, $K=1.1$, $B=15$, $T=5$).

The highest average precision is obtained on the manual transcriptions (R1: 0.5411), but as already observed on our development results the performance degradation using speech recognizer outputs is fairly modest (2% and 3% for the HTK and LIMSIR automatic transcriptions). Comparing Tables 2 and 3, it can be observed that the gain using PBRF for query expansion is smaller on the SDR'99 data set than it was on the SDR'98 data set. This is may be linked to the choice of the epoch for the PBRF corpus or to a suboptimal tuning of the BRF parameters.

ADDITIONAL RESULTS

In this section some post-evaluation experiments with the Okapi-based system are reported. We also report here some of the development experiments comparing the Okapi and Markovian approaches.

Adjusting System Parameters

Having no experience with IR system tuning before this evaluation, we found it rather difficult to properly set the Okapi parameters (K and b) and the query expansion parameters (B and T), so as to maximize the average precision for both the R1 and S1 conditions on the SDR'98 test set with the associated 23 queries.

Extensive experiments were carried out to investigate the IR performance for a range of parameter values. Figures 2 through 4 show the effect of the Okapi parameters (b and K) on the average precision for SDR'98-R1, SDR'99-R1

and SDR'99-S1 respectively, using a baseline system without query expansion. The iso-data lines of the resulting surfaces are shown, along with their projections on the base plane which highlights the location of the extrema.

Figures 5 through 7 show the effect of the BRF parameters (B and T) on the average precision for SDR'98-R1, SDR'99-R1 and SDR'99-S1 respectively, using the system with query expansion based on both BRF and PBRF.

It is clear from these plots that the best parameter settings for the SDR'99 data set cannot be easily predicted from the SDR'98 results. However it was clearly possible to choose better BRF parameter values than those resulting from our development work. In particular too few terms are kept (i.e. the T value was really underestimated). New results using $T=10$ (which corresponds to the best results on the SDR'98-S1 data) are given in Table 4 (label *cw* for the Okapi term weighting).

Markovian term weighting

As a natural extension of our work on speech recognition relying on Markovian assumptions for both acoustic and language modeling, we investigated a term weighting function based on a simple query/document model in place of the Okapi formula. A comparable approach has been previously employed with success [11, 15]. Assuming a unigram model, the following term weighting is used:

$$mw_{t,d} = qtf_t \times \log(\alpha \Pr(t|d) + (1 - \alpha) \Pr(t)). \quad (3)$$

Table 4 gives the results for both Okapi (*cw*) and Markovian (*mw*) term weightings on the SDR'99 data set with the following parameter settings: $b=0.86$, $K=1.1$, $B=15$, $T=10$, $\alpha=0.5$. In both cases query expansion relies on the term of-fer weight defined above. It can be seen that very comparable results can be achieved using the two term weighting schemes.

<i>data</i>	<i>meth.</i>	<i>base</i>	<i>brf</i>	<i>pbrf</i>	<i>brf+pbrf</i>
98-R1	<i>cw</i>	0.4689	0.5648	0.5591	0.5786
	<i>mw</i>	0.4695	0.5936	0.5574	0.5889
98-S1	<i>cw</i>	0.4594	0.5118	0.5621	0.5761
	<i>mw</i>	0.4558	0.5121	0.5884	0.5745
99-R1	<i>cw</i>	0.4711	0.5318	0.5147	0.5487
	<i>mw</i>	0.4691	0.5354	0.5098	0.5430
99-S1	<i>cw</i>	0.4327	0.5239	0.4919	0.5350
	<i>mw</i>	0.4412	0.5302	0.4943	0.5398

Table 4: Comparison of IR results on the SDR'98 and SDR'99 data sets using both Okapi and Markovian term weightings ($b=0.86$, $K=1.1$, $B=15$, $T=10$, $\alpha=0.5$). R1: reference transcript. S1: automatic speech transcription.

Okapi settings

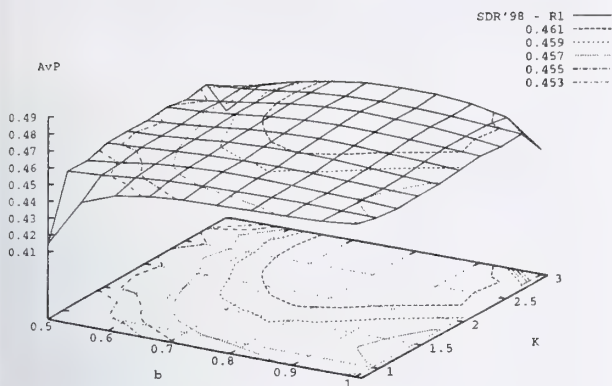


Figure 2: Plot of average precision vs Okapi parameters b and K for the baseline system (no query expansion), SDR'98 - R1. (Best AveP is 0.4836 for $b=0.80$ and $K=2.5$)

Blind Relevance Feedback

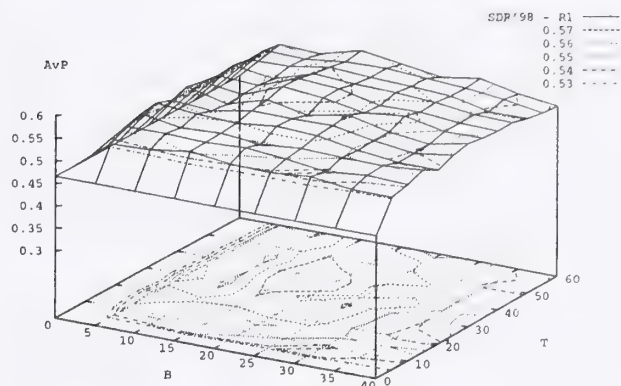


Figure 5: Plot of average precision vs BRF parameters B and T for BRF+PBRF query expansion, SDR98 - R1. (Best AveP is 0.5835 for $B=15$ and $T=35$).

Okapi settings

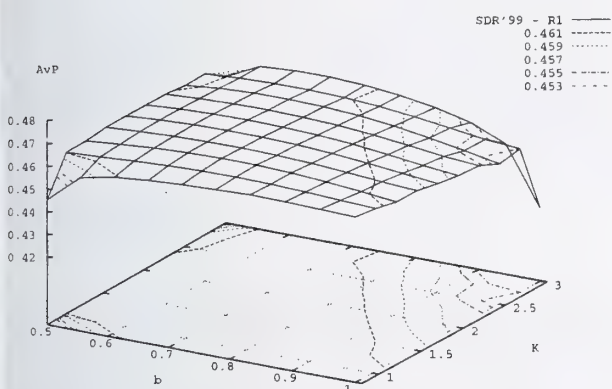


Figure 3: Plot of average precision vs Okapi parameters b and K for the baseline system (no query expansion), SDR'99 - R1. (Best AveP is 0.4736 for $b=0.75$ and $K=1.3$).

Blind Relevance Feedback

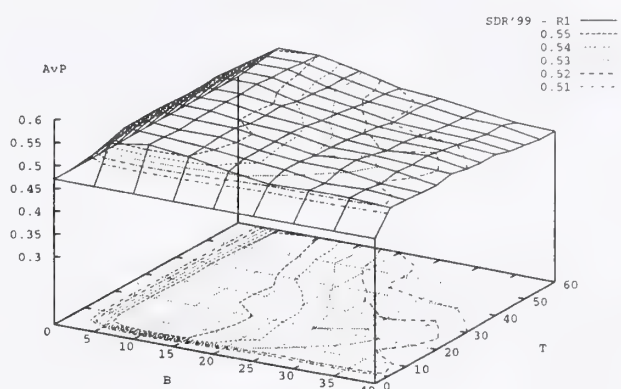


Figure 6: Plot of average precision vs BRF parameters B and T for BRF+PBRF query expansion, SDR99 - R1. (Best AveP is 0.5615 for $B=5$ and $T=40$).

Okapi settings

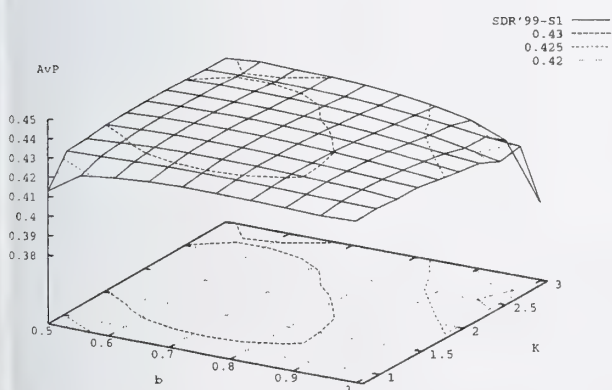


Figure 4: Plot of average precision vs Okapi parameters b and K for the baseline system (no query expansion), SDR'99 - S1. (Best AveP is 0.4401 for $b=0.65$ and $K=2.1$).

Blind Relevance Feedback

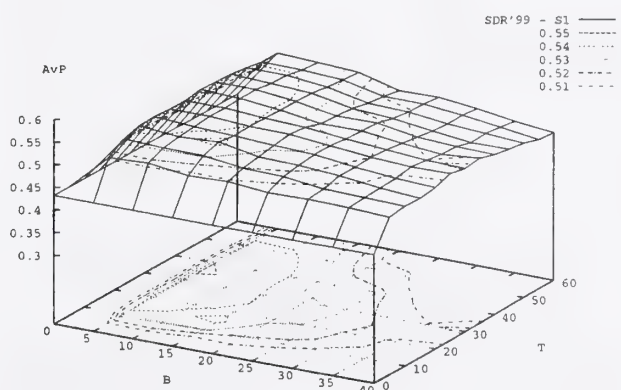


Figure 7: Plot of average precision vs BRF parameters B and T for BRF+PBRF query expansion SDR99 - S1. (Best AveP is 0.5515 for $B=5$ and $T=35$).

SUMMARY & DISCUSSION

In this paper we have presented our complete SDR'99 system, and highlighted our development work. This system was built by combining an adapted version of the LIMSI 1998 Hub-4E transcription system for speech recognition with an IR system based on the Okapi term weighting function. The transcription system achieved a word error of 21.5% measured on a 10h subset of the SDR'99 data set. Using the parameter settings optimized on the SDR'98 data set, average precision of 0.5636 and 0.5072 respectively were obtained on the SDR'98 and SDR'99 data sets using the transcriptions produced by the LIMSI recognizer. These values are quite close to the average precisions obtained on manual transcripts, indicating that the transcription quality is not the limiting factor on IR performance. Our post-evaluation experiments indicate that (unfortunately) the evaluation settings for the BRF were suboptimal.

ACKNOWLEDGMENTS

This work has been partially financed by the European Commission and the French Ministry of Defense. The authors gratefully acknowledge the participation of Michèle Jardino, Remi Lejeune and Patrick Paroubek to this work.

REFERENCES

- [1] <http://www.thomson.com/psmedia/bnews.html>
- [2] <ftp://ciir-ftp.cs.umass.edu/pub/stemming/>
- [3] J.L. Gauvain, G. Adda, L. Lamel and M. Adda-Decker, "Transcribing Broadcast News: The LIMSI Nov96 Hub4 System," *ARPA Speech Recognition Workshop*, Chantilly, VA, pp. 56-63, February 1997.
- [4] J.L. Gauvain, L. Lamel and G. Adda, "The LIMSI 1997 Hub-4E Transcription System," *DARPA Broadcast News Transcription & Understanding Workshop*, Landsdowne, VA, pp. 75-79, February 1998.
- [5] J.L. Gauvain, L. Lamel and G. Adda, "Partitioning and Transcription of Broadcast News Data," *Proc. ICSLP'98*, 5, pp. 1335-1338, Sydney, December 1998.
- [6] J.L. Gauvain, L. Lamel, G. Adda and M. Adda-Decker, "Transcription of Broadcast News", *Proc. ESCA EuroSpeech'97*, Rhodes, pp. 907-910, September 1997.
- [7] J.L. Gauvain, L. Lamel, M. Adda-Decker, "The LIMSI Nov93 WSJ System," *Proc. ARPA Spoken Language Technology Workshop*, Princeton, NJ, pp. 125-128, March 1994.
- [8] J.L. Gauvain and C.H. Lee, "Maximum *a Posteriori* Estimation for Multivariate Gaussian Mixture Observation of Markov Chains," *IEEE Trans. on Speech and Audio Processing*, 2(2), pp. 291-298, April 1994.
- [9] J.L. Gauvain, L. Lamel, G. Adda and M. Jardino, "The LIMSI 1998 HUB-4E Transcription System," *Proc. DARPA Broadcast News Workshop*, Herndon, VA, pp. 99-104, February 1999.
- [10] A.G. Hauptmann, M. Witbrock and M. Christel, "News-on-Demand-'An Application of Informedia Technology'," *Digital Libraries Magazine*, September 1995.
- [11] D. Hiemstra and K. Wessel, "Twenty-One at TREC-7: Ad-hoc and Cross-language track," *NIST Special Publication 500-242: The Seventh Text REtrieval Conference (TREC-7)*, NIST, Gaithersburg, MD, November 1998.
- [12] S. E. Johnson, P. Jurlin, G. L. Moore, K. Spärk Jones and P. C. Woodland, "Spoken document retrieval for TREC-7 at Cambridge University", *NIST Special Publication 500-242: The Seventh Text REtrieval Conference (TREC-7)*, NIST, Gaithersburg, MD, November 1998.
- [13] F. de Jong, J.L. Gauvain, J. de Hartog and K. Netter, "OLIVE: Speech Based Video Retrieval," *Proc. CBMI'99*, Toulouse, October 1999.
- [14] C.J. Leggetter and P.C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Computer Speech & Language*, 9(2), pp. 171-185, 1995.
- [15] D. Miller, T. Leek and R. Schwartz, "Using Hidden Markov Models for Information Retrieval," *NIST Special Publication 500-242: The Seventh Text REtrieval Conference (TREC-7)*, NIST, Gaithersburg, MD, November 1998.
- [16] M. F. Porter, "An algorithm for suffix stripping", *Program*, 14, pp. 130-137, 1980.
- [17] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu and M. Gatford, "Okapi at TREC-3", *NIST Special Publication 500-226: Overview of the Third Text REtrieval Conference (TREC-3)*, NIST, Gaithersburg, MD, November 1994.
- [18] K. Spärk Jones, S. Walker and S. E. Robertson, "A probabilistic model of information retrieval: development and status," *a Technical Report of the Computer Laboratory, University of Cambridge, U.K.*, 1998.
- [19] S. Walker and R. de Vere, "Improving subject retrieval in on-line catalogues: 2. Relevance feedback and query expansion," *British Library Research Paper 72*, British Library, London, U.K., 1990.
- [20] P.C. Woodland, T. Neielar and E. Whittaker, "Language Modeling in the HTK Hub5 LVCSR," presented at the 1998 Hub5E Workshop, September 1998.

Kenney Ng

Spoken Language Systems Group
MIT Laboratory for Computer Science
545 Technology Square, Cambridge, MA 02139 USA

ABSTRACT

In this paper we present a novel probabilistic information retrieval model that scores documents based on the relative change in the document likelihoods, expressed as the ratio of the conditional probability of the document given the query and the prior probability of the document before the query is specified. The document likelihoods are computed using statistical language modeling techniques and the model parameters are estimated automatically and dynamically for each query to optimize well-specified (maximum likelihood) objective functions. We derive the basic retrieval model, describe the details of the model, and present some extensions to the model including a method to perform automatic feedback. Development experiments are performed using the TREC-6 ad hoc text retrieval task and performance is measured using the TREC-7 ad hoc task. Official evaluation results on the 1999 TREC-8 ad hoc task are also reported. The performance results demonstrate that the model is competitive with current state-of-the-art retrieval approaches.

1. INTRODUCTION

Probabilistic modeling for information retrieval (IR) has a long history [3]. Many of these approaches try to evaluate the probability of a document being relevant (R) to a given query Q by estimating $p(R|Q, D_i)$ for every document D_i in the collection. These relevance probabilities are then used to rank order the retrieved documents. However, due to the imprecise definition of the concept of relevance and the lack of available relevance training data, reliably estimating these probabilities has been a difficult task. Because of the the nature of the IR task, training data in the form of document-query pairs labeled with their corresponding relevance judgments is not generally available *a priori*. Previously seen queries, for which relevance information can be created, can be used for training but their applicability to new queries is not clear. Some relevance information can be obtained in a multi-pass retrieval strategy by using relevance feedback. However, only a small number of relevance judgments is typically generated. Many of these probabilistic methods are better suited for related applications, such as information filtering, where more relevance training data is available [6, 7].

Instead of the imprecisely defined notion of relevance, we consider the better defined measure of likelihood. In particular, we examine the relative change in the likelihood of a document before and after a query is specified, and use that as the metric for scoring and ranking the documents. The idea is that documents that become more likely after the query is specified are probably more useful to the user and should score better and be ranked ahead of those documents whose likelihoods either stay the same or decrease. The document likelihoods are computed using statistical language modeling techniques and the model parameters are estimated automatically and dynamically for each query to optimize well-specified objective functions.

The paper is organized as follows. In Section 2, we derive the basic retrieval model, describe the details of the model, and present some extensions to the model including a method to perform automatic feedback. We also discuss some related modeling approaches. Next, in Section 3, we evaluate the performance of the retrieval model

and present experimental results on the TREC-6 ad hoc text retrieval task. Then, in Section 4, we objectively evaluate the system on the TREC-7 ad hoc task and report official evaluation results on the 1999 TREC-8 ad hoc task. Finally, we close with a summary in Section 5.

2. INFORMATION RETRIEVAL MODEL

Given a collection of n documents, $\{D_i\}_{i=1}^n$, each document D_i has a prior likelihood given by $p(D_i)$. After a query Q is specified by a user, the likelihood of each document changes and becomes that given by the conditional probability: $p(D_i|Q)$. Some documents will become more likely after the query is specified while others will either remain the same or become less likely. The documents that become more likely are probably more useful to the user and should score better and be ranked ahead of those that either stay the same or become less likely. As a result, we propose to use the relative change in the document likelihoods, expressed as the likelihood ratio of the conditional and the prior probabilities, as the metric for scoring and ranking the documents in response to query Q :

$$S(D_i, Q) = \frac{p(D_i|Q)}{p(D_i)} \quad (1)$$

We can decompose this likelihood ratio score into more easily estimated components using Bayes' Rule and rewrite (1) as:

$$S(D_i, Q) = \frac{p(Q|D_i)p(D_i)/p(Q)}{p(D_i)} = \frac{p(Q|D_i)}{p(Q)} \quad (2)$$

where $p(Q|D_i)$ is the probability of query Q given document D_i and $p(Q)$ is the prior probability of query Q . Each document D_i specifies a different language model Λ_i . We can view $p(Q|D_i)$ as the probability that query Q is generated by Λ_i , the language model associated with document D_i . This means that our goal during the retrieval process is to find those documents in the collection that maximize the likelihood of the query. These documents should be the ones that are most useful to the user who specified query Q .

The $p(Q)$ term represents the probability that query Q is generated from a document independent (general) language model Λ , and serves as a normalization factor. Since $p(Q)$ is constant for all documents D_i given a specific query Q , it does not affect the ranking of the documents and can be safely removed from the scoring function. However, this $p(Q)$ normalization factor is useful if we want a meaningful interpretation of the scores (as a relative change in the likelihood) and if we want to be able to compare scores across different queries. In Section 3.3, we illustrate the usefulness of $p(Q)$ for these purposes. In addition, the $p(Q)$ normalization factor is an important part of the automatic feedback extension to the basic model as we will see in Section 2.2. For these reasons, we will keep the $p(Q)$ term in the scoring function in (2).

2.1. Model Details

In order to compute the score in (2), we need to be able to estimate the quantities $p(Q|D_i)$ and $p(Q)$. To do this, we make the assumption that the query Q is drawn from a multinomial distribution over the set of possible terms in the corpus and document D_i specifies

the parameters of the multinomial model. This gives us the following estimates for $p(Q|D_i)$ and $p(Q)$:

$$p(Q|D_i) = \frac{n!}{\prod_{t=1}^k c_t!} \prod_{t=1}^k p(t|D_i)^{c_t} \quad (3)$$

$$p(Q) = \frac{n!}{\prod_{t=1}^k c_t!} \prod_{t=1}^k p(t)^{c_t} \quad (4)$$

where c_t is the number of times term t occurs in query Q , k is the number of distinct terms in the corpus, $n = \sum_{t=1}^k c_t$ is the total number of terms in query Q , $p(t|D_i)$ is the probability of query term t occurring in document D_i with the constraint $\sum_{t=1}^k p(t|D_i) = 1$, and $p(t)$ is the probability of query term t occurring in the document collection with the constraint $\sum_{t=1}^k p(t) = 1$. Substituting (3) and (4) into (2) and simplifying (noting that $c_t! = 1$ for $c_t = 0$), we have:

$$S(D_i, Q) = \prod_{t=1}^k \left(\frac{p(t|D_i)}{p(t)} \right)^{c_t} \quad (5)$$

Since $x^0 = 1$ for all x , the product over all k terms can be replaced by a product over only the terms that occur in the query:

$$S(D_i, Q) = \prod_{t \in Q} \left(\frac{p(t|D_i)}{p(t)} \right)^{c_t} \quad (6)$$

To simplify computation and to prevent numerical underflows, we perform the score computation in the log domain:

$$S_l(D_i, Q) = \log S(D_i, Q) = \sum_{t \in Q} c_t \log \left(\frac{p(t|D_i)}{p(t)} \right) \quad (7)$$

We note that since the logarithm is a monotonic transformation, the rank ordering of the documents using the log score remains the same as that using the original score.

In the original multinomial model, c_t is the number of times term t occurs in query Q and can only take on integral values: $c_t = 0, 1, \dots, n$. We would like to generalize c_t so that it can take on non-negative real values. This will allow more flexible weighting of the query terms including the use of fractional counts which will be useful in our automatic relevance feedback extension (Section 2.2) and query section weighting (Section 3.6). To indicate this generalization in the scoring function, we replace c_t in (7) with $q(t)$, which can be interpreted as the weight of term t in query Q :

$$S_l(D_i, Q) = \sum_{t \in Q} q(t) \log \left(\frac{p(t|D_i)}{p(t)} \right) \quad (8)$$

This generalization does not affect the ranking of the documents since it is equivalent to adding a query-dependent constant multiplicative factor, $1/n$, to the score in (7) to convert the c_t counts to the $q(t)$ numbers. In fact, we can interpret $q(t)$ as $p(t|Q)$, the probability of term t occurring in query Q , if $q(t) = c_t/n$ where $n = \sum_t c_t$.

We note that the scoring function in (8) can be related to the Kullback-Leibler distance [2], which is an information theoretic measure of the divergence of two probability distributions $p_1(x)$ and $p_2(x)$:

$$KL(p_1(x), p_2(x)) = - \sum_x p_2(x) \log \left(\frac{p_1(x)}{p_2(x)} \right) \quad (9)$$

To show this relationship, we start by rewriting (8) as follows:

$$S_l(D_i, Q) = \sum_{t \in Q} q(t) \log p(t|D_i) - \sum_{t \in Q} q(t) \log p(t) \quad (10)$$

Next, we add in and subtract out $\sum_{t \in Q} q(t) \log q(t)$, rearrange terms, and then collapse terms to get:

$$S_l(D_i, Q) = \underbrace{\sum_{t \in Q} q(t) \log \left(\frac{p(t|D_i)}{q(t)} \right)}_{-KL(q(t), p(t|D_i))} - \underbrace{\sum_{t \in Q} q(t) \log \left(\frac{p(t)}{q(t)} \right)}_{+KL(q(t), p(t))} \quad (11)$$

Recall that $q(t)$ can be interpreted as $p(t|Q)$, the probability of term t in query Q , $p(t|D_i)$ is the probability of term t in document D_i , and $p(t)$ is the probability of term t in the general language (i.e., using a document-independent model). The first term in (11) is the (negative) KL divergence between the term distribution of query Q and document D_i . If the two term distributions are identical, then the divergence will be zero. As the difference between the query and document distributions becomes greater, the divergence increases, and the score decreases (because of the negative sign on the term). The second term is the KL divergence between the term distribution of query Q and a general document-independent model. Since this term doesn't depend on the document, it has no effect on the rankings of the retrieved documents; it only serves as a bias or normalization factor. It is query-dependent and only comes into play if we compare scores across different queries.

We also note that the scoring function in (8) has the form of the standard vector space model. It consists of the sum over all terms t in the query of the product of a query dependent factor, $q(t)$, and a document dependent factor, $\log(p(t|D_i)/p(t))$. It turns out that many probabilistic models can be expressed in the standard vector space model format [3, 9, 15]. The models differ in what the query and document factors are and how they are estimated.

Next, we need to estimate the probabilities $p(t|D_i)$ and $p(t)$. We start by considering their maximum likelihood (ML) estimates:

$$p_{ml}(t|D_i) = \frac{d_i(t)}{\sum_{t=1}^k d_i(t)} \quad (12)$$

$$p_{ml}(t) = \frac{\sum_{i=1}^n d_i(t)}{\sum_{i=1}^n \sum_{t=1}^k d_i(t)} \quad (13)$$

where $d_i(t)$ is the number of occurrences of term t in document D_i , k is the number of distinct terms in the corpus, and n is the number of documents in the collection.

With a large document collection, there is enough data for $p_{ml}(t)$ to be robustly estimated. However, this ML estimate will assign a probability of zero to terms that do not occur in the document collection. To avoid this undesirable property, we can use Good-Turing (GT) methods to estimate $p(t)$ [10]. GT methods provide probability estimates for both observed and unobserved terms with the constraint that the total probability of all terms must sum to one. For unobserved terms, GT methods provide an estimate of the *total* probability of these terms. This total probability can then be divided among the possible unobserved terms to provide per term probability estimates. For observed terms, GT methods provide probability estimates for these terms that are consistent with estimating non-zero probabilities for the unobserved terms. This is done by reducing the total probability of the observed terms to be less than one. Good-Turing methods work as follows. If a certain term t occurs r times in the document collection, the ML estimate of $p(t)$ is given by:

$$p_{ml}(t) = r/N \quad (14)$$

where N is the total number of terms observed in the document collection. With GT estimation, the count r is replaced by a modified count r^* which is calculated as:

$$r^* = (r + 1) \frac{N_{r+1}}{N_r} \quad (15)$$

where N_r is the number of terms that occurs exactly r times in the document collection. As a result, the GT estimate of $p(t)$ for observed terms is given by:

$$p_{gt}(t) = p_r = r^*/N \quad (16)$$

where $N = \sum_r r N_r$ is the total number of terms observed in the document collection. The GT estimate for the *total* probability of unobserved terms is given by:

$$p_0 = N_1/N \quad (17)$$

This total probability is then divided equally among the possible unobserved terms to provide per term probability estimates. Using the observed N_r values to calculate r^* in (15) can become problematic if $N_r = 0$ for some r . As a result, it is necessary to pre-smooth N_r so that it never equals zero. There are many different possible smoothing methods and each gives rise to a slightly different GT approach. We use the Simple Good-Turing (SGT) approach described in [5]. Basically N_r is linearly smoothed (in the log domain) and a decision rule is used to decide when to switch from using the observed N_r values to the smoothed values.

Unlike the estimate for $p(t)$, the quantity $p_{ml}(t|D_i)$ is likely to be poorly estimated regardless of the size of the document collection because of the limited size of the individual documents. Many of the terms in the model will have zero probability. There are many different ways to compensate for this sparse data problem. One approach is to model the term distributions using parametric distributions such as Beta and Dirichlet distributions. A standard statistical language modeling approach, and the one we adopt, is to linearly interpolate the more detailed $p_{ml}(t|D_i)$ model with a better estimated, but more general model, for example, $p_{gt}(t)$ [10]:

$$p(t|D_i) = \alpha p_{ml}(t|D_i) + (1 - \alpha) p_{gt}(t) \quad (18)$$

where α is the mixture weight. The estimate-maximize (EM) algorithm [4] can be used to estimate α to maximize the (log) likelihood of query Q given document D_i :

$$\alpha^* = \arg \max_{\alpha} \log(p(Q|D_i)) \quad (19)$$

$$= \arg \max_{\alpha} \sum_{t \in Q} q(t) \log(\alpha p_{ml}(t|D_i) + (1 - \alpha) p_{gt}(t)) \quad (20)$$

In the above formulation, there is a different α for each document D_i . To simplify the model and to provide more data for parameter estimation, we can "tie" the α weight across the documents so that there is only a single, document-independent, α for each query Q . The following iterative procedure can then be used to estimate α :

1. Initialize α to a random estimate between 0 and 1.
2. Update α using:

$$\alpha' = \frac{1}{\sum_{t \in Q} \sum_{i \in \mathcal{I}_Q} q(t)} \times \sum_{t \in Q} \sum_{i \in \mathcal{I}_Q} q(t) \frac{\alpha p_{ml}(t|D_i)}{\alpha p_{ml}(t|D_i) + (1 - \alpha) p_{gt}(t)}$$

3. If α has converged (i.e., $|\alpha' - \alpha| < \delta$ for some small threshold δ) then stop. Otherwise, set $\alpha = \alpha'$ and goto step 2.

In this procedure, \mathcal{I}_Q contains the indices of the set of documents used to estimate α for query Q . We need to decide which documents should be in this set. If we use *all* the documents in the collection (i.e., $\mathcal{I}_Q = \{1, \dots, n\}$), the query terms will occur so seldomly in the entire collection that α will almost always be set to zero. That would not be very useful. What we want is a reasonable estimate of α for those documents that are likely to be relevant to the query since

they are the ones that we are interested in. Ideally, we want the set of documents to be those that *are* relevant to query Q . However, since this information is not available, we need to use an approximation. One approach is to borrow the technique used in automatic relevance feedback [15] (see Section 2.2). Basically, we perform a preliminary retrieval run using an initial guess for α (e.g., $\alpha = 0.5$) and assume that the top M retrieved documents are relevant to the query. These M top-scoring documents then become the set we use to estimate the α weight for query Q . $M = 5$ is a typical value that we use.

Using the approach described above, a separate α is estimated for each query Q . If desired, one can pool the query terms across all the queries and estimate a single query-independent α . It is important to note that the above procedure estimates the mixture parameters dynamically using the current query and the current document collection. This is in contrast to the standard approach of determining static, query-independent, model parameter values by empirically tuning on an old development set which typically consists of a different set of queries and potentially a different collection of documents. In Section 3.4, we explore the effect of different estimated α values on retrieval performance and examine query-specific and query-independent α 's.

In summary, the final metric used for scoring document D_i in response to query Q is obtained by substituting the estimates for $p(t)$ and $p(t|D_i)$ (Equations 16 and 18, respectively) into (8):

$$S_i(D_i, Q) = \sum_{t \in Q} q(t) \log \left(\frac{\alpha p_{ml}(t|D_i) + (1 - \alpha) p_{gt}(t)}{p_{gt}(t)} \right) \quad (21)$$

2.2. Automatic Relevance Feedback

Automatic relevance feedback is a proven method for improving information retrieval performance [6]. The process works in three steps. First, the original query is used to perform a preliminary retrieval run. Second, information from these retrieved documents are used to automatically construct a new query. Third, the new query is used to perform a second retrieval run to generate the final results. A commonly used query reformulation strategy, the Rocchio algorithm [15], starts with the original query, Q , then adds terms found in the top N_t retrieved documents and subtracts terms found in the bottom N_b retrieved documents to come up with a new query, Q' . Modifying the query in this way adds new terms that occur in documents that are likely to be relevant to the query and eliminates terms that occur in documents that are probably non-relevant. The query terms are also reweighted. The goal is to improve the ability of the query to discriminate between relevant and non-relevant documents.

We extend our basic retrieval model to include an automatic relevance feedback processing stage by developing a new query reformulation algorithm that is specific to our probabilistic model. Recall that in our retrieval model, we score document D_i in response to query Q using the likelihood ratio score (2):

$$S(D_i, Q) = \frac{p(Q|D_i)}{p(Q)} \quad (22)$$

Since the documents are ranked based on descending values of this score, we can view the goal of the automatic feedback procedure as trying to create a new query Q' (based on the original query Q and the documents retrieved from the preliminary retrieval pass) such that the score using the new query is better than the score using the original query for those documents D_i that are relevant to the query:

$$\frac{p(Q'|D_i)}{p(Q')} \geq \frac{p(Q|D_i)}{p(Q)} \quad \text{for } i \in \mathcal{I}_Q \quad (23)$$

Because \mathcal{I}_Q , the set of relevant documents for query Q , is not known, we use an approximation and assume that the top scoring documents from a preliminary retrieval run using the original query are relevant.

There are many different ways to decide which of the top scoring documents to select. One approach is to simply select a fixed number, M , of the top scoring documents. One concern with this approach is that the selected documents can have very disparate scores. There can be a big score difference between the first and the M^{th} document. Another approach is to use an absolute score threshold, θ , so only documents with scores above θ are selected. With this approach, it is possible to not have any documents that score above the threshold. A different approach, and the one we adopt, is to use a relative score threshold, $\gamma \leq 1$, so documents that score within a factor of γ of the top scoring document are selected:

$$\text{select } D_i \text{ if } \frac{S(D_i, Q)}{\max_{D_i} S(D_i, Q)} \leq \gamma \quad (24)$$

This approach results in a variable number of documents for each query, but the selected documents will have similar scores. A typical threshold value that we use is $\gamma = 0.75$.

Since we want to improve the score for all the documents in the set \mathcal{I}_Q simultaneously, we need to deal with the set of documents jointly. One way to do this is to create a new joint document D' by pooling together all the documents in the set \mathcal{I}_Q so the number of occurrences of term t in the joint document D' is given by:

$$d'(t) = \sum_{i \in \mathcal{I}_Q} d_i(t) \quad (25)$$

Another variation is to weight the contribution of each document, D_i , by its preliminary retrieval score, $S(D_i, Q)$, so documents that score better have more of an impact:

$$d'(t) = \sum_{i \in \mathcal{I}_Q} S(D_i, Q) d_i(t) \quad (26)$$

Using this new joint document, D' , the inequality in (23) becomes:

$$\frac{p(Q'|D')}{p(Q')} \geq \frac{p(Q|D')}{p(Q)} \quad (27)$$

Substituting our models for the conditional and prior probabilities and working in the log domain (Equation 8), we have

$$\sum_{t \in Q'} q'(t) \log \left(\frac{p(t|D')}{p(t)} \right) \geq \sum_{t \in Q} q(t) \log \left(\frac{p(t|D')}{p(t)} \right) \quad (28)$$

Let us consider the creation of the new query Q' in two steps. First, let us examine which terms should be *removed* from the original query Q in order to improve the score. Second, we can then examine which terms from the joint document D' should be *added* to the query to further improve the score.

Starting with the original query Q , we consider each query term t and determine whether it should be included or excluded from the new query Q' . Since the query term weights $q(t)$ are constrained to be greater than zero, the only way that a query term t can decrease the score is if $\frac{p(t|D')}{p(t)} < 1$. Therefore, if we exclude such terms from the new query Q' (while keeping the term weights the same, i.e., $q'(t) = q(t)$), we can be assured that the inequality in (28) is satisfied. This selection criteria makes intuitive sense since it basically states that query terms that occur more frequently in the general collection than in the pooled document D' (which is created from assumed relevant documents) should not be used.

Next, we consider which terms from the joint document D' should be included to the query Q' in order to further improve the score. Following the same arguments as those used above, and noting that $q'(t) > 0$, we see that only terms t for which $\frac{p(t|D')}{p(t)} > 1$ can increase

the score. As a result, we will only add those terms from D' that satisfy this property. Using this term selection criteria, we maintain the inequality in (28) with each newly included term. Substituting the estimates for $p(t)$ and $p(t|D_i)$ (Equations 16 and 18, respectively), the term selection criteria becomes:

$$\frac{p(t|D')}{p(t)} > 1 \quad (29)$$

$$\frac{\alpha p_{\text{ml}}(t|D') + (1 - \alpha) p_{\text{gt}}(t)}{p_{\text{gt}}(t)} > 1$$

$$\frac{p_{\text{ml}}(t|D')}{p_{\text{gt}}(t)} > 1 \quad (30)$$

Therefore, we can equivalently use $\frac{p_{\text{ml}}(t|D')}{p_{\text{gt}}(t)} > 1$ or $\log \left(\frac{p_{\text{ml}}(t|D')}{p_{\text{gt}}(t)} \right) > 0$ to perform the term selection.

The only issue that remains is the estimation of appropriate values for the weights $q'(t)$ of the newly included query terms. Since the value of the score can be increased arbitrarily by using increasingly larger values of $q'(t)$, we need to constrain the aggregate value of the weights. One reasonable constraint is that the magnitude of the query weights be unity:

$$\|Q'\| = \sqrt{\sum_{t \in Q'} q'(t)^2} = 1 \quad (31)$$

Adopting this constraint, we can use the technique of Lagrange multipliers [1] to find the set of query term weights, $\{q'(t)\}$, that maximizes the score:

$$\sum_{t \in Q'} q'(t) \log \left(\frac{p(t|D')}{p(t)} \right) \quad (32)$$

The corresponding Lagrangian function is given by:

$$\mathcal{L}(Q', \lambda) = \sum_{t \in Q'} q'(t) \log \left(\frac{p(t|D')}{p(t)} \right) + \lambda \left(\sqrt{\sum_{t \in Q'} q'(t)^2} - 1 \right) \quad (33)$$

Taking the partial derivative of (33) with respect to λ and setting it to zero, we get back the constraint equation:

$$\frac{\partial}{\partial \lambda} \mathcal{L}(Q', \lambda) = 0 \quad (34)$$

$$\sqrt{\sum_{t \in Q'} q'(t)^2} = 1 \quad (35)$$

Taking the partial derivative of (33) with respect to the query term weight $q'(t)$ and setting it to zero, we get

$$\frac{\partial}{\partial q'(t)} \mathcal{L}(Q', \lambda) = 0 \quad (36)$$

$$\log \left(\frac{p(t|D')}{p(t)} \right) + \lambda \frac{q'(t)}{\sqrt{\sum_{t \in Q'} q'(t)^2}} = 0 \quad (37)$$

Taking the second derivative, we get

$$\frac{\partial^2}{\partial q'(t)^2} \mathcal{L}(Q', \lambda) = \lambda (1 - q'(t)^2) \quad (38)$$

For the score to be maximized, we need this second derivative to be less than zero. Since $0 < q'(t) < 1$, we must have $\lambda < 0$ in order for (38) to be negative.

Combining equations (35) and (37) and solving for $q'(t)$, we get

$$q'(t) = -\frac{1}{\lambda} \log \left(\frac{p(t|D')}{p(t)} \right) \quad (39)$$

Since we require $\lambda < 0$, we see that the appropriate query weights simply have to be proportional to their score contribution:

$$q'(t) \propto \log \left(\frac{p(t|D')}{p(t)} \right) \quad (40)$$

This weighting scheme makes intuitive sense since we want to emphasize terms that contribute more to the score. If desired, we can determine the exact value of the proportionality factor by substituting (39) back into (35) and solving for λ . Doing this, we find that:

$$\lambda = -\sqrt{\sum_{t \in Q'} \left(\log \left(\frac{p(t|D')}{p(t)} \right) \right)^2} \quad (41)$$

Our description of the automatic relevance feedback procedure is now complete. We have a procedure that automatically creates a new query Q' based on the original query Q and a set of top-ranked documents retrieved from a preliminary retrieval pass. The goal of the procedure is to increase the likelihood ratio scores of the top-ranked documents by removing certain terms from the original query and adding new terms from the top-ranked documents with appropriate term weights. Hopefully improving the scores will lead to improved information retrieval performance.

We note that this automatic feedback procedure significantly increases the number of terms in the query since many of the terms in the joint document D' will satisfy the selection criteria (29). If desired, one can limit the number of additional terms by modifying this term selection criteria so only terms with scores greater than some threshold $\phi \geq 1$ will be included:

$$\text{add term } t \text{ if } \frac{p(t|D')}{p(t)} > \phi \quad (42)$$

In Section 3.5, we examine the ability of this automatic relevance feedback procedure to improve retrieval performance and explore the effects of limiting the number of new query terms by increasing the value of ϕ in (42).

2.3. Related Work

In our retrieval model, we use the relative change in the likelihood of a document D_i before and after the user query Q is specified, expressed as the likelihood ratio of the conditional and the prior probabilities, $\frac{p(D_i|Q)}{p(D_i)}$, as the metric for scoring and ranking the documents. A document that becomes more likely after the query is specified is probably more useful to the user than one that either remains the same or becomes less likely. This score can be equivalently rewritten as $\frac{p(Q|D_i)}{p(Q)}$. Since we need to estimate $p(Q|D_i)$, the probability of query Q given document D_i , our model is related to several recently proposed IR approaches which also make use of this probabilistic quantity [9, 11, 12].

In [12] and [9], a language modeling argument is used to directly posit that $p(Q|D_i)$ is an appropriate quantity for scoring document D_i in response to query Q . Mixture models are then used to compute this quantity. In [11], the probability that document D_i is relevant given query Q , $p(D_i \text{ is } R|Q)$, is used to score the documents. This quantity can be rewritten, using Bayes Rule, as $\frac{p(Q|D_i \text{ is } R)p(D_i \text{ is } R)}{p(Q)}$. A generative hidden Markov model (HMM) is then used to compute the quantity $p(Q|D_i \text{ is } R)$.

Although our retrieval model shares this commonality with these other approaches, there are some important differences. First, as described above, our model is derived starting from a different theoretical justification. Second, different modeling assumptions and estimation techniques are used to determine the underlying probabilistic quantities. Although we use the standard technique of mixture models to estimate the quantity $p(Q|D_i)$, the underlying probabilistic components in our mixture model are different from those used in [12] and [9]. We back-off to the term's probability of occurrence in the entire document collection. In [9], the back-off is to the term's document frequency while in [12] the back-off is a scaled version of the term's mean probability of occurrence in documents that contain the term. We also automatically estimate the mixture model parameters dynamically (for each query Q) to maximize the likelihood of the query given a set of top scoring documents $\{D_i\}$ from the current document collection. This is in contrast to the standard approach of determining static, query-independent, mixture model parameter values by empirically tuning on an old development set. In addition, we attempt to deal with unobserved query terms in a more principled way by using Good-Turing techniques to smooth the underlying probability models. Finally, we develop a new automatic relevance feedback strategy that is specific to our probabilistic model. The procedure automatically creates a new query (based on the original query and a set of top-ranked documents from a preliminary retrieval pass) that optimizes a well-specified objective function. In particular, the term selection and the term weight estimation procedures are designed to maximize the likelihood ratio scores of the set of documents presumed to be relevant to the query. Hopefully, improving these scores will lead to improved retrieval performance.

3. INFORMATION RETRIEVAL EXPERIMENTS

Our information retrieval model is evaluated on the TREC-6, TREC-7, and TREC-8 ad hoc text retrieval tasks [6–8]. The ad hoc task involves searching a static set of documents using new queries and returning an ordered list of documents ranked according to their relevance to the query. The retrieved documents are then evaluated against relevance assessments created for each query.

Retrieval performance is measured in terms of a tradeoff between *precision* and *recall*. Precision is the number of relevant documents retrieved over the total number of documents retrieved. Recall is the number of relevant documents retrieved over the total number of relevant documents in the collection. Because it may be cumbersome to compare the performance of different systems using precision-recall curves, a single number performance measure called *mean average precision* (mAP) is commonly used [6]. It is computed by averaging the precision values at the recall points of all relevant documents for each query and then averaging those across all the test set queries.

In this section, we briefly describe the data corpus that comprise the TREC-6, TREC-7, and TREC-8 tasks, mention the text preprocessing that was done, and then present several retrieval experiments using the TREC-6 task. In these development experiments, we explore the usefulness of the $p(Q)$ normalization in the scoring, the effect of using different mixture weights in the probability model, the use of the automatic relevance feedback processing, and section-based weighting of the query terms.

3.1. Data Corpus

The document collection in the TREC-6, TREC-7, and TREC-8 ad hoc retrieval tasks consists of text stories from various news and information sources. Details of the composition and size of the collections are given in Table 1. The documents in the TREC-7 task are a subset of those in the TREC-6 task (documents from the *Congressional Record* are excluded from the TREC-7 collection). The document collection used in the TREC-8 task is identical to that used in TREC-7. Each collection contains approximately 2 gigabytes of text from over half a million documents.

Data Set	Size (MB)	# docs	Avg. # wrds/doc
<i>Financial Times</i> (FT)	564	210,158	412.7
<i>Federal Register</i> (FR)	395	55,630	644.7
<i>Congressional Record</i> (CR)	235	27,922	1373.5
<i>FBIS</i> (FBIS)	470	130,471	543.6
<i>L.A. Times</i> (LA)	475	131,896	526.5
TREC-6 (all sources)	2139	556,077	541.9
TREC-7 (4 sources: no CR)	1904	528,155	497.9
TREC-8 (same as TREC-7)	1904	528,155	497.9

Table 1: Statistics for the document collections used in the TREC-6, TREC-7, and TREC-8 ad hoc retrieval tasks.

Data Set (topic #'s)	# of Words		
	Min	Max	Avg.
TREC-6 (301-350)	47	156	88.4
title	1	5	2.7
description	5	62	20.4
narrative	17	142	65.3
TREC-7 (351-400)	31	114	57.6
title	1	3	2.5
description	5	34	14.3
narrative	14	92	40.8
TREC-8 (401-450)	23	98	51.3
title	1	4	2.4
description	5	32	13.8
narrative	14	75	35.1

Table 2: Statistics for the test topics used in the TREC-6, TREC-7, and TREC-8 ad hoc retrieval tasks. There are 50 topics in each retrieval task.

There are 50 queries (also called “topics”) for each of the TREC-6, TREC-7, and TREC-8 ad hoc retrieval tasks. Topic numbers 301-350 are used in the TREC-6 task, while 351-400 are used in the TREC-7 task, and 401-450 are used in the TREC-8 task. Each topic consists of three sections: a title, a description, and a narrative. Statistics regarding the size of the topics are shown in Table 2.

In order to evaluate the performance of a retrieval system, relevance assessments must be provided for each topic. In other words, for each topic in the test set, the set of the known relevant documents in the collection needs to be determined. Since there are too many documents for complete manual inspection, an approximate method, known as the “pooling method,” is used to find the set of relevant documents [7]. For each topic, a pool of possible relevant documents is first created by taking the top 100 documents retrieved from the various participating systems. Next, each document in this pool is manually assessed to determine its relevance. Finally, those documents that are judged relevant become the “answers” for the topic and are used to conduct the performance evaluations. Summary statistics for the

Data Set (topic #'s)	# of Relevant Docs			
	Min	Max	Avg.	Total
TREC-6 (301-350)	3	474	92.2	4611
TREC-7 (351-400)	7	361	93.5	4674
TREC-8 (401-450)	6	347	94.6	4728

Table 3: Statistics for the number of relevant documents for the topics in the TREC-6, TREC-7, and TREC-8 ad hoc retrieval tasks. There are 50 topics in each retrieval task.

number of relevant documents for the topics in the TREC-6, TREC-7, and TREC-8 ad hoc tasks are shown in Table 3. We note that there is great variability. Some topics have many relevant documents while others have only a few.

In our retrieval experiments, we use the TREC-6 task as the “development” data set for tuning and optimizing our retrieval model. Most of the contrasting experiments will be done on the TREC-6 task. We reserve the TREC-7 task for use as the “test” data to objectively test our final retrieval model. An official TREC “evaluation” run was done using the TREC-8 task. Following standard practices, we use the entire topic statement (consisting of the title, description, and narrative components) in our retrieval experiments, unless otherwise noted.

3.2. Text Preprocessing

Before a document is indexed, it undergoes a relatively standard set of text preprocessing steps. First, the text is normalized to remove non-alphanumeric characters like punctuation and to collapse case. Next, sequences of individual characters are automatically grouped to create single terms in an “automatic acronym aggregation” stage. For example, the text string “U. S. A.” would be converted to “u s a” after normalization and then to “usa” after acronym aggregation. Stop words, derived from a list of 600 words, are then removed from the document. In addition to standard English function words, certain words frequently used in past TREC topics such as “document,” “relevant,” and “irrelevant” are also included in the list. Finally, the remaining words are conflated to collapse word variants using an implementation of Porter’s stemming algorithm [13]. To maintain consistency, each topic description also undergoes the exact same text preprocessing steps before it is indexed and used to retrieve documents from the collection.

3.3. $p(Q)$ Normalization

As discussed in Section 2.1, the $p(Q)$ normalization factor in the scoring function (2) does not affect the ranking of the documents because it is constant for all documents D_i given a specific topic Q . However, we choose to keep this factor because it helps to provide a meaningful interpretation of the scores as a relative change in the likelihood and allows the document scores to be more comparable across different topics. In addition, as we’ve seen in Section 2.2, the $p(Q)$ normalization factor plays an important role in the term selection and weighting stages of the automatic relevance feedback procedure.

To illustrate the difference between the (unnormalized) likelihood score ($p(Q|D_i)$) and the (normalized) likelihood ratio score ($\frac{p(Q|D_i)}{p(Q)}$), Figure 1 plots the distribution of these two scores for the subset of relevant documents for the 50 topics (topics 301-350) in the TREC-6 task. The likelihood scores have a very wide distribution across queries while the likelihood ratio scores are more tightly clustered. Box plots are used to indicate the score distributions. The center line in the box indicates the mean value while the lower and upper edges of the box indicate, respectively, the lower and upper quartiles. The vertical lines extending below and above the box show the entire range of the scores. We observe that the document likelihood scores can differ drastically depending on the topic. The best score for some topics (e.g., 309 and 316) are worse than the lowest scores for other topics (e.g., 315 and 339). Scoring the documents using the likelihood ratio puts the scores for the different topics on a much more comparable range. These scores can be interpreted as how much more likely the document has become after the topic is specified than before.

In the computation of the standard information retrieval measures of recall, precision, and mean average precision (mAP), each topic is treated independently. Precision-recall curves are generated for each topic separately using individual thresholds. These separate curves are then combined to create an aggregate precision-recall curve and the single number mAP measure. Since document scores are not compared across the different topics in the computation of these standard information retrieval measures, they will be identical for both the like-

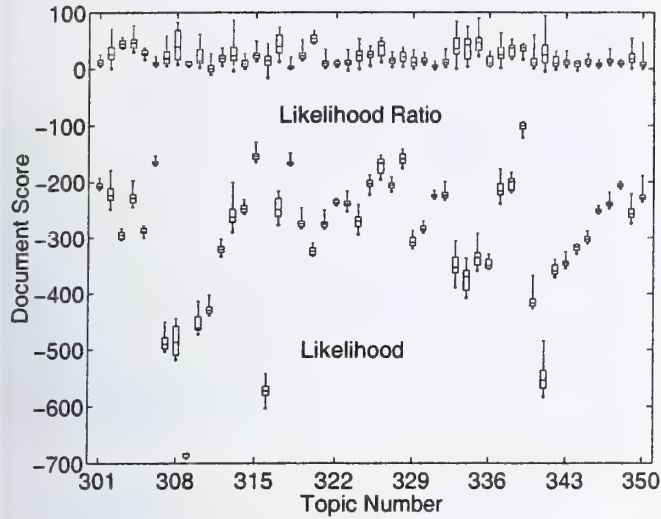


Figure 1: Distribution of likelihood and likelihood ratio scores for the relevant documents for topics 301-350 in the TREC-6 task.

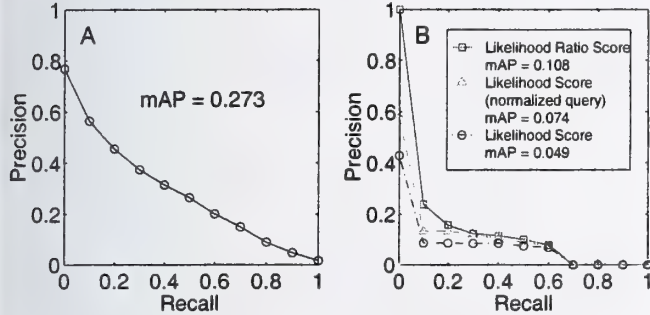


Figure 2: (A) Precision-Recall curve and mean average precision (mAP) score on the TREC-6 ad hoc task using a mixture weight of $\alpha = 0.5$. (B) Precision-Recall curves resulting from using a single threshold across all topics on the TREC-6 data for three different scoring methods.

likelihood and likelihood ratio scores. In Figure 2A, we plot the resulting aggregate precision-recall curve and mean average precision (mAP) measure on the TREC-6 ad hoc task for the 50 topics (301-350). This is the baseline performance of our retrieval model using the preliminary retrieval run and a fixed topic-independent mixture weight of $\alpha = 0.5$. A performance of $\text{mAP}=0.273$ is achieved.

There are certain related applications, such as document clustering and topic detection, where it is important to be able to compare document scores across different “topics.” To quantify how much the likelihood ratio score can help in these situations, we can generate a precision-recall curve that results from using a *single threshold* across all the different topics. In this way, we can measure the ability of the different scoring methods to handle across topic score comparisons. In Figure 2B, we show such recall-precision curves and the associated mAP measure for the 50 topics on the TREC-6 ad hoc data using three different scoring methods. As expected, the raw likelihood score performs poorly when cross topic score are compared. A normalized likelihood score (normalized by the number of the terms in the topic) gives slightly better results. However, the likelihood ratio score, which is not only normalized by the number of terms in the topic but also by the prior likelihoods of the terms, gives even better performance.

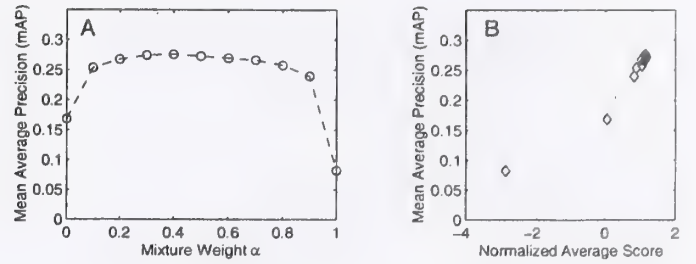


Figure 3: (A) Retrieval performance in mean average precision (mAP) on the TREC-6 task as a function of the value of the mixture weight α . (B) Scatter plot of mAP versus the normalized average score of the top documents for each of the different α weights.

Mixture Weight Estimate	mAP
Fixed ($\alpha = 0.5$)	0.273
Topic-Independent ($\alpha = 0.434$)	0.275
Topic-Dependent (variable α)	0.278

Table 4: Retrieval performance in mean average precision (mAP) on the TREC-6 task using different estimates of the mixture weight α .

3.4. Mixture Weights

In this section, we explore the effect of different α mixture weight estimates on retrieval performance and examine topic-specific and topic-independent α ’s. To quantify the sensitivity of the model to the mixture weight α , we explore a range of possible weight values and measure the resulting retrieval performance. In Figure 3A, we plot retrieval performance in mean average precision (mAP) on the TREC-6 ad hoc task as a function of the value of the mixture weight α . We see that although retrieval performance does vary with the value of α , there is a relatively large range of stable and good performance.

A scatter plot of mAP versus the normalized average score of the top retrieved documents for each of the different α weights is shown in Figure 3B. The plot shows that retrieval performance is well correlated ($\rho = 0.96$) with the document scores. This means that we can use the document scores to find an appropriate value of α that can be expected to give reasonably good retrieval performance. In fact, the automatic α parameter estimation procedure that we described in Section 2.1 tries to maximize the likelihood of topic Q given document D_i , $p(Q|D_i)$, which is the numerator of the document score (2). Since the denominator of the score, $p(Q)$, remains unchanged, this is equivalent to maximizing the entire document score. As shown in Table 4, running the preliminary retrieval pass using a fixed weight of $\alpha = 0.5$ results in a retrieval performance of $\text{mAP}=0.273$. Performance improves slightly to $\text{mAP}=0.275$ when we use the automatically estimated topic-independent weight of $\alpha = 0.434$.

Since topic statements can be very different from one another, we can expect that using the same α weight for every topic is probably suboptimal. This is indeed the case as illustrated in Figure 4, which plots retrieval performance in average precision (AP) for three different topics (327, 342, and 350) from the TREC-6 ad hoc task as a function of the value of the mixture weight α . We see that the optimal value of α for each topic can be very different. To address this issue, we can estimate topic-dependent α ’s, as discussed in Section 2.1. In Figure 5, we plot the distribution of the automatically estimated topic-dependent α mixture weights for the 50 topics (301-350) in the TREC-6 task. Many of the weights are centered around the topic-independent estimated value of $\alpha=0.434$ but there are several topics that have weights at the extreme ends of the range. Using these topic-dependent α mixture weights, retrieval performance is further improved to $\text{mAP}=0.278$ as shown in the last row of Table 4.

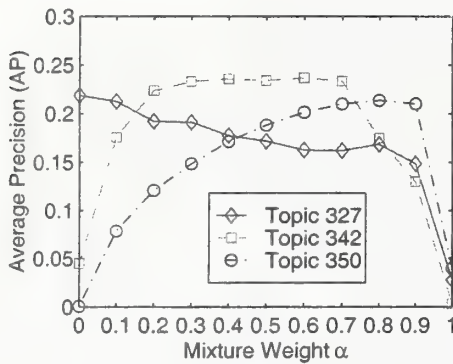


Figure 4: Retrieval performance in average precision (AP) for topics 327, 342, and 350 from the TREC-6 task as a function of the value of the mixture weight α .

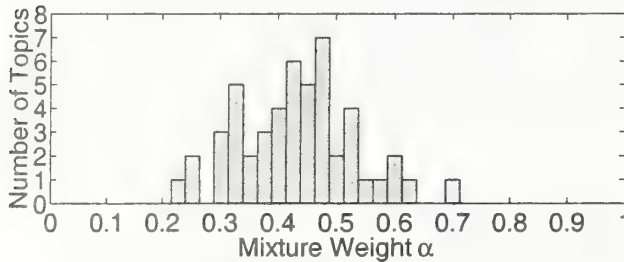


Figure 5: Distribution of the automatically estimated topic-dependent α mixture weights for topics 301-350 in the TREC-6 task. The pooled α is 0.434 and the average α is 0.432.

3.5. Automatic Feedback

In this section, we evaluate the automatic relevance feedback procedure described in Section 2.2 and examine its ability to improve retrieval performance. Recall that during the feedback process, a new topic Q' is created by removing certain terms from the original topic Q and adding new terms (with appropriate term weights) from the top scoring documents obtained from a preliminary retrieval run. The number of new terms added to Q' can be controlled by changing the threshold ϕ in the term selection criteria (42). Lowering the value of ϕ adds more terms. Note that new query terms are added in order of decreasing contribution to the total score; terms that contribute most to improving the score are added first.

Figure 6 plots retrieval performance, measured in mean average precision (mAP), on the TREC-6 ad hoc task as the number of terms in the new topic Q' is varied. Running the preliminary retrieval pass using the original topics, which average 27 unique terms each, gives a performance measure of mAP=0.273. Using automatic feedback to modify the topic results in significant performance improvements as illustrated in Figure 6. As more terms are included in the new topic Q' , performance improves sharply, reaches a maximum at around 250-300 terms, declines slightly, and then levels off. The retrieval performance peaks at mAP=0.317 for approximately 250 terms.

It is interesting to note that performance is relatively stable over a wide range of topic sizes spanning 200 to 700 terms. By significantly increasing the number of terms in the topic, one may expect that the topic specification may become too broad and, as a result, the retrieval performance will be adversely affected. However, this does not happen in our case because the terms added to the new topic Q' are weighted proportionally to their score contribution as specified in (40). As a result, many of the additional terms will only have a small effect on the total score.

In terms of determining an appropriate ϕ threshold to use, one

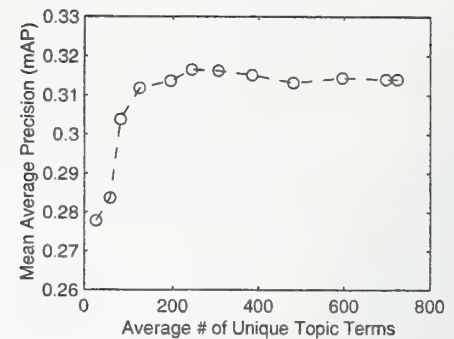


Figure 6: Retrieval performance in mean average precision (mAP) on the TREC-6 ad hoc task using the automatic feedback procedure as the number of terms in the new topic Q' is varied. By lowering the threshold ϕ in the term selection criteria (42), more terms are included in the new topic.

possibility is to simply set $\phi = 1.0$ so all terms that contribute positively to the score will be included. This corresponds to adding the maximum number of terms allowed by our procedure. Using this threshold value on the TREC-6 ad hoc task, the average number of unique terms in the new query Q' grows to 724.2. However, from the behavior shown in Figure 6, the same or even slightly better performance can be achieved by using many fewer terms. We find empirically that a reasonable threshold to use is $\phi = 0.25 \times S_{\max}(D_i, Q)$, where $S_{\max}(D_i, Q)$ is the score of the top retrieved document D_i for topic Q . This relative threshold value puts us in the stable performance region without adding too many terms to the new topic Q' .

We conclude that incorporating the automatic feedback processing stage into the retrieval system significantly improves retrieval performance. Large gains of 0.035 to 0.04 in absolute mean average precision (from mAP=0.278 to 0.317) are obtained.

3.6. Topic Section Weighting

As described in Section 3.1, the queries or topics statements for the retrieval tasks consist of three different sections: a title, a description, and a narrative. We can expect that the different sections contain different amounts of useful information. To quantify how useful each section is in finding the relevant documents for the topic, we can evaluate the retrieval performance resulting from using each topic section individually. In Table 5, we show retrieval performance in mean average precision (mAP) on the TREC-6 ad hoc task using the different topic sections. We examine the use of the title, description, and narrative sections individually, the title and description sections combined (T+D), and all three sections together (T+D+N). Retrieval performance after the preliminary and feedback retrieval stages are shown along with the average number of unique terms in each topic section. We can make several observations. First, the different topic sections vary greatly in their size. The title, description, and narrative sections average 2.5, 8.8, and 21.7 unique terms, respectively. Second, even though the title section contains the fewest terms, its preliminary retrieval performance is better than that of the other two sections. This implies that the terms from the title section are more useful than those from the other sections. Third, using multiple topic sections results in better performance. Combining the title and description (T+D) gives performance that is better than any of the individual sections, and using all three (T+D+N) gives even better performance. Fourth, automatic feedback improves performance in all cases but is more effective when there are more terms in the topic statement. In particular, the gain for the title section is small compared to the gains for the other sections.

In the above experiments, when we combined the different topic sections, we weighted each section equally. This means that in the

Topic Section	Avg # Unique Topic Terms	mAP	
		Preliminary	Feedback
Title (T)	2.5	0.225	0.230
Description (D)	8.8	0.178	0.221
Narrative (N)	21.7	0.218	0.253
T+D	9.5	0.247	0.296
T+D+N (All)	27.0	0.278	0.317

Table 5: Retrieval performance in mean average precision (mAP) on the TREC-6 ad hoc task using different sections of the topics: title, description, and narrative individually, title and description combined (T+D), and all three sections together (T+D+N). The second column shows the average number of unique terms in each section. The third and fourth columns show performance after the preliminary and feedback retrieval stages, respectively.

Topic Section	mAP	
	Preliminary	Feedback
T+D	0.247	0.296
T+D (weighted)	0.260	0.297
T+D+N	0.278	0.317
T+D+N (weighted)	0.303	0.325

Table 6: Retrieval performance in mean average precision (mAP) on the TREC-6 ad hoc task with and without topic section weighting. Performance is shown for two different topic configurations: title and description combined (T+D), and all three sections (title, description, and narrative) together (T+D+N). Performance after the preliminary and feedback retrieval stages are shown.

T+D+N case which combines all three sections, the title section only contributes, on average, 2.5 terms to the combined topic while the narrative section contributes 21.7 terms. From the performance of the individual topic sections in Table 5, it is clear that the terms in the title section are more useful than those in the narrative section. Maybe emphasizing terms from some sections (e.g., the title), more than terms from other sections (e.g., the narrative) in the formation of the combined topic will result in better performance than just equally weighting all the sections. This is indeed the case. In [11], they found that weighting the topic terms based on what section they are in improved retrieval performance. In [14], the output from several retrieval runs using the individual topic sections are combined to give improved performance.

We can adopt a similar approach of weighting terms based on their topic section membership to try to further improve retrieval performance. One method is to weight the terms from each topic section in proportion to the average score of the top documents retrieved using that section. The idea is that topic sections that give higher document scores should be emphasized more than those that give lower scores. We are basically using the document score as a predictor of retrieval performance which is consistent with our retrieval model which ranks documents based on descending values of the document scores. Because the scores are normalized (likelihood ratios), we are able to compare them across different topic statements (consisting of different topic sections) to determine which topic formulation is better. Basically, we run three retrieval passes using the title, description, and narrative sections individually, compute the average score of the top retrieved documents from each run, and then use those scores in weighting the terms from the different topic sections. The process used to select the set of top scoring documents is the same as the one used in the automatic feedback procedure (24). For each new task, this procedure is used to automatically determine the appropriate section weights. Using this topic section weighting scheme on the

Topic Section	mAP	
	Preliminary	Feedback
T+D	0.212	0.243
T+D+N (All)	0.250	0.284

Table 7: Retrieval performance in mean average precision (mAP) on the TREC-7 ad hoc task using different topic specifications: title and description combined (T+D), and all three sections together (T+D+N). Performance for the preliminary and automatic feedback retrieval stages are shown.

TREC-6 ad hoc task, we get section weights of 4.2 for the title, 1.8 for the description, and 1.0 for the narrative. This weighting emphasizes the title section the most, then the description section, and finally the narrative section.

Weighting the topic sections in this way results in a small but consistent performance improvement over weighting each section equally, as shown in Table 6. Retrieval performance in mean average precision (mAP) on the TREC-6 ad hoc task with and without topic section weighting is shown for two different topic configurations: title and description combined (T+D), and all three sections (title, description, and narrative) together (T+D+N). The effect of the topic section weighting is greater on the preliminary retrieval pass than on the automatic feedback pass. Recall that the feedback process already includes term selection and term weighting. As a result, some of the gains from the section weighting may already be accounted for in the feedback processing.

4. INFORMATION RETRIEVAL PERFORMANCE

All of the above experiments were conducted on the TREC-6 ad hoc text retrieval task. These development experiments were used to configure the system and to tune some system parameters. The final retrieval system has the following configuration:

- Dynamic (for each query) and automatic estimation of the mixture parameter α using the procedure described in Section 2.1 with the following parameter: $M=5$.
- Use of the second pass automatic relevance feedback procedure described in Section 2.2 with the following parameters: $\gamma=0.75$ (Equation 24) and $\phi = 0.25 \times S_{\max}(D_i, Q)$ (Equation 42), where $S_{\max}(D_i, Q)$ is the score of the top retrieved document D_i for topic Q .
- Use of the query section weighting procedure described in Section 3.6 with the following parameter: $\gamma=0.75$ (Equation 24). The section weights are automatically determined for each new set of test queries.

Now that the system configuration is set, we need to evaluate the performance of the final retrieval system on new sets of held-out test data. We use the TREC-7 and TREC-8 ad hoc retrieval tasks, described in Section 3.1, for this purpose.

4.1. Retrieval Performance on the Test Set

In Table 7, we show the performance (in mAP) of our system on the TREC-7 ad hoc task. Retrieval is done using two types of topics: one consisting of the title and description sections only (T+D) and the other consisting of all three (title, description, and narrative) sections (T+D+N). Performance is shown for the preliminary retrieval pass and the automatic feedback pass. We observe that automatic feedback significantly improves performance for all conditions and that using longer topic statements is better. The performance level of mAP=0.284 on this task is competitive with the performance of the state-of-the-art retrieval systems on the identical task as reported in the TREC-7 conference [7].

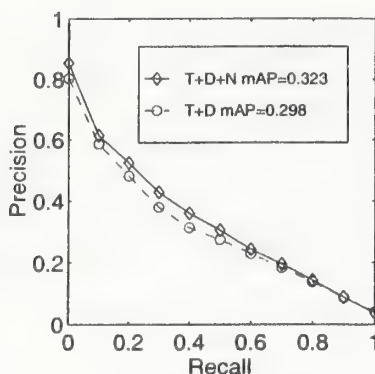


Figure 7: Precision-Recall curves for the TREC-8 ad hoc task. Performance using topics consisting of title and description (T+D), and full topics consisting of the title, description, and narrative sections (T+D+N) are shown.

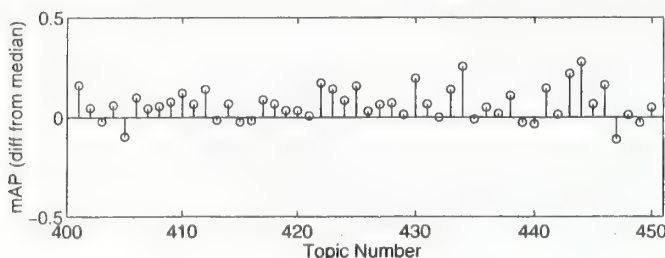


Figure 8: Difference (in mean average precision) from the median for each of the 50 topics in the TREC-8 ad hoc task. Full topics consisting of the title, description, and narrative sections are used.

4.2. Retrieval Performance on the Evaluation Set

We participated in the 1999 TREC-8 ad hoc text retrieval evaluation [8]. Performance on the official TREC-8 ad hoc task using our probabilistic retrieval model is shown in Figure 7. Two retrieval runs were submitted: one consisting of the title and description sections only (T+D) and the other consisting of all three (title, description, and narrative) sections (T+D+N). A performance of $mAP=0.298$ is achieved using the shorter topics; the full topics gave a $mAP=0.323$. Out of the 55 participating systems that used the short topic description, our system ranked sixth behind systems that had mAP s of 0.321, 0.317, 0.317, 0.306, and 0.301. Out of the 37 participating systems that used the entire topic description, our system ranked fourth behind systems that had mAP s of 0.330, 0.324, and 0.324. Difference in mAP from the median performance for each of the 50 topics for the full topic run (T+D+N) are shown in Figure 8. Of the 50 topics, 40 scored at or above the median level and seven achieved the maximum score. On this task, we again see that our retrieval model is very competitive with current state-of-the-art retrieval systems.

5. SUMMARY

In this paper we present a novel probabilistic information retrieval model and demonstrate its capability to achieve state-of-the-art performance on large standardized text collections. The retrieval model scores documents based on the relative change in the document likelihoods, expressed as the ratio of the conditional probability of the document given the query and the prior probability of the document before the query is specified. Statistical language modeling techniques are used to compute the document likelihoods and the model parameters are estimated automatically and dynamically for each query to optimize well-specified maximum likelihood objective functions. An automatic relevance feedback strategy that is specific to the proba-

bilistic model is also developed. The procedure automatically creates a new query (based on the original query and a set of top-ranked documents from a preliminary retrieval pass) by selecting and weighting query terms so as to maximize the likelihood ratio scores of the set of documents presumed to be relevant to the query. To benchmark the performance of the new retrieval model, we use the standard ad hoc text retrieval tasks from the TREC-6 and TREC-7 text retrieval conferences. Official evaluation results on the 1999 TREC-8 ad hoc text retrieval task are also reported. Experimental results indicate that the model is able to achieve performance that is competitive with current state-of-the-art retrieval approaches.

6. ACKNOWLEDGMENTS

This research was performed under the supervision of Dr. Victor W. Zue and was supported by DARPA under contract N66001-96-C-8526 and monitored through NCCOSC.

7. REFERENCES

- [1] D. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Cambridge, MA: Academic Press, 1982.
- [2] T. Cover and J. Thomas, *Elements of Information Theory*. New York: John Wiley and Sons, 1991.
- [3] F. Crestani, M. Lalmas, C. J. V. Rijsbergen, and I. Campbell, "Is this document relevant?...Probably: A survey of probabilistic models in information retrieval," *ACM Computing Surveys*, vol. 30, no. 4, pp. 528-552, Dec. 1998.
- [4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1-38, 1977.
- [5] W. A. Gale and G. Sampson, "Good-Turing frequency estimation without tears," *Journal of Quantitative Linguistics*, no. 2, pp. 217-237, 1995.
- [6] D. K. Harman, ed., *Sixth Text REtrieval Conference (TREC-6)* Gaithersburg, MD, USA, National Institute for Standards and Technology, 1997. NIST-SP 500-240.
- [7] D. K. Harman, ed., *Seventh Text REtrieval Conference (TREC-7)* Gaithersburg, MD, USA, National Institute for Standards and Technology, 1998. NIST-SP 500-242.
- [8] D. K. Harman, ed., *Eighth Text REtrieval Conference (TREC-8)* Gaithersburg, MD, USA, National Institute for Standards and Technology, 1999. NIST-SP.
- [9] D. Hiemstra and W. Kraaij, "Twenty-one at TREC-7: Ad-hoc and cross-language track," in *Seventh Text REtrieval Conference (TREC-7)*, Gaithersburg, MD, USA, 1998. NIST-SP 500-242.
- [10] F. Jelinek, *Statistical Methods for Speech Recognition*. Cambridge, MA: MIT Press, January 1999.
- [11] D. Miller, T. Leek, and R. Schwartz, "BBN at TREC-7: Using hidden markov models for information retrieval," in *Seventh Text REtrieval Conference (TREC-7)*, Gaithersburg, MD, USA, 1998. NIST-SP 500-242.
- [12] J. Ponte and W. B. Croft, "A language modeling approach to information retrieval," in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 275-281, 1998.
- [13] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130-137, July 1980.
- [14] S. Robertson, S. Walker, and M. Beaulieu, "Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive," in *Seventh Text REtrieval Conference (TREC-7)*, Gaithersburg, MD, USA, 1998. NIST-SP 500-242.
- [15] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York: McGraw-Hill, 1983.

High Selectivity and Accuracy with READWARE's Automated System of Knowledge Organization

By Tom Adi, O. K. Ewell and Patricia Adi
Management Information Technologies, Inc. (MITi)¹

Email: mitioke@readware.com

October 27, 1999

Abstract

READWARE performs a fully automatic text analysis that implements a system of knowledge organization based on *knowledge types*. A knowledge type is a set of instructions that identifies a set of *knowledge elements* in any text. Knowledge types include *concepts* (word sets), *topics* (an expandable hierarchical scheme of common knowledge types spanning politics, business, health, and so on), *probes* (investigative knowledge types), *issues* (knowledge types used in decisionmaking) and *document subjects* (traditional classification of documents by themes). An MITi analyst used this system to translate TREC topic specifications into highly selective queries (few hits per query) in two adhoc runs with high relevance rates (2019 / 3060 hits in the READWARE run and 2774 / 5785 hits in the READWARE2 run).

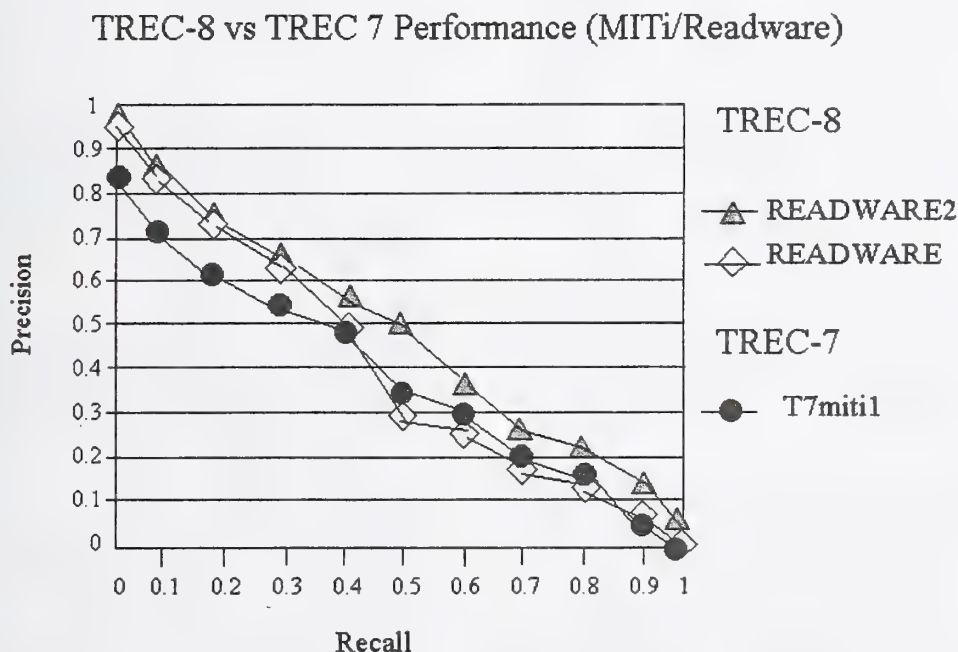


Figure 1.

¹ Management Information Technologies, Inc. has been developing software for automatic text analysis and search based on a system of knowledge organization since 1985. MITi's technology is marketed under the trade name **READWARE®**. The product line includes **ConSearch** for Windows workstations, the **IpServer** for internet/intranet applications and the Readware Software Development Kit for custom solutions.

1. Introduction

MITi is participating in the TREC for the second time with its READWARE technology. We used our product **ConSearch** to perform two manual adhoc runs. The adhoc task consists of finding the documents relevant to fifty specified topics in a pool of more than half a million documents.

READWARE is a text analysis technology based on a system of knowledge organization consisting of **knowledge types**. A knowledge type is a set of instructions (usually a set of queries) that identifies a certain set of **knowledge elements** in any text. **Concepts** are a network of basic knowledge types. A concept is a set of words that are seen as strongly related knowledge elements. A **superconcept** is a compound knowledge type consisting of several closely related concepts. Concepts and superconcepts are inspired by terms from ancient languages. READWARE offers the following advanced knowledge types:

1) **Document Subjects.** Document subjects reflect the traditional classification of documents by themes. Analysts can create their own **user subjects**.

2) **Probes.** Probes are investigative knowledge types such as *who/where, why, how often, success, growth* and *roots*.

3) **Issues.** Issues are knowledge types used in decisionmaking such as *trends, emerging needs, potential trouble, checking on those in charge* and *clash of interests*.

4) **Topics.** Topics form an expandable hierarchical scheme of common knowledge types. Analysts can easily construct their own **user topics**. The current scheme con-

sists of the following topic areas that contain a total of 336 topics:

Current READWARE Topic Areas

Way of Life	World View
	Laws & Lawmakers
	Courts
	Those in Charge
Communities & Relations	Family
	Countries & Regions
	International
Basic Needs	Safety and Security
	Health
	Knowledge & Technology
	Sources of Information (Media)
	Environment
	Housing
	Transportation
	Food
	Energy
	Business
Culture & Recreation	Culture
	Recreational Activities
	Travel
	Sports

READWARE also offers Boolean logic, sequence enforcement, context sizing, word search, phrase search and document-level search.

This year, we refined and extended our knowledge types, especially topics and probes. The number of topics doubled since last year. We also included many British spellings and terms.

There are three basic search strategies: word search, concept search and superconcept search. The document pool may be limited by specifying which document subjects are desired or undesired. Users may mix strategies using a different strategy for every query item. A variable-size sliding search window scans each document for certain words, phrases, concepts, superconcepts, topics, issues and probes. The window size (context size) can be set in the

query to values ranging from one tenth the query size to 20 times the query size.

READWARE hits must have a complete set of semantic relations with the query. We ranked the hit documents by the complexity of the queries used. The more items and positions the generating queries contained, the higher the hit rank. But unlike last year, a hit document was ranked higher if there were multiple hit spots or a hit spot was in the headline or near the top of the document. READWARE highlights the exact hit spots in the documents.

2. Data Preparation

Knowledge type implementations are stored in the **ConceptBase** (5 MB for English). The READWARE text analysis module automatically locates all knowledge elements in the texts before the analyst sits down to make queries.

We used a Pentium III (450 MHz) with 256 MB of RAM and a 12 Gigabyte disk. A fully automatic data preparation (text analysis) took about 13 hours of CPU time.

TREC 8 files were decompressed and their end-of-line sequences were optimized using a utility program. The READWARE text analysis module then split the files in memory into documents using the <DOC> and <DOCNO> tags. This was done without physical duplication by keeping track of document lengths and their positions in the original files. Our default tag filter made sure that tags were not analyzed. The text parts between the <subject> and </subject> tags were also not analyzed.

The READWARE analyst module scanned every document to determine the positions of names (non-concept words), concepts, topics, issues and probes and to identify

document subjects. Locating the knowledge elements belonging to all READWARE knowledge types meant asking over a million queries to every document using a variable-size sliding analysis window.

Analysis results were stored in 3 files:

docs._ (68.5 MB): vital document info (document file, subject, issues and topics)

sigs._ (1.04 GB): signature database (positions of names and knowledge element in all documents)

optdx._ (176 MB): optimized index

3. Query Construction

One MITi analyst used READWARE's knowledge types (a few thousand concepts, a 336-topic hierarchy, 27 probes, 21 issues and 54 document subjects) and the advanced search features (Boolean logic, sequence enforcement, context sizing, word search, phrase search and document-level search) to refine and perfect the TREC topic specifications (title, description and narrative) and turn them into a more consistent and complete set of automatically executable queries.

READWARE indicates to the analyst what concept a certain word in the TREC topic specification belongs to so that she can search for the full concept rather than a single word if she chooses to do so. And whenever the specifications are ambiguous or incomplete, the analyst can navigate the knowledge type schemes and follow her hunches. She can expand her thoughts and fill in the gaps by testing queries made with indicated concepts, topics from the same topic area, similar probes, related issues and other document subjects. Extensive word expansion and thought expansion are already

implicit in the knowledge type implementations.

The MITi analyst first constructed 65 *user topics*. These are READWARE topics that should be found in every document dealing with the TREC topic. Used as queries, they identify a baseline pool of documents for every TREC topic.

For example, here are the TREC specification and the READWARE user topics for TREC topic 420:

TREC Specification for Topic 420

Title: Carbon monoxide poisoning
Description: How widespread is carbon monoxide poisoning on a global scale?
Narrative: Relevant documents will contain data on what carbon monoxide poisoning is, symptoms, causes, and/or prevention. Advertisements for carbon monoxide protection products or services are not relevant. Discussions of auto emissions and air pollution are not relevant even though they can contain carbon monoxide.

User Topic 367: Carbon Monoxide

```
//The following line is the topic title
=aa carbon monoxide (367)
//Look for the words "carbon" and "monoxide"
//The phrase "emission" (from narrative) not in the context
//Topic 106 (air pollution, from narrative) not in the context
!"emission"
carbon monoxide !T:106
```

User Topic 373: CO

```
//The following line is the topic title
=aa CO (373)
//Look for the phrase "CO" (case-sensitive)
//The words emission (from narrative), dioxide and funder
//should not be in the context
/"CO"
/+ !emission !dioxide !funder
```

The lines starting with "/" in the user topic boxes are comment lines that explain the queries and their relationship to the TREC topic specification. The phrase *air pollution* in the description is a READWARE topic title (topic 106).

In addition to the queries in the user topics, the analyst formulated 771 queries, an average of 14 queries per TREC topic (compared to 18 last year). She made queries by combining baselines or user topics with knowledge types and phrases related to the TREC topic specifications. Topic 445 (women clergy) needed the least number of queries, just two. Topic 401 (foreign minorities, Germany) required the most number of queries, 65.

To satisfy different styles of judging, the analyst made two runs. In the first run (labeled READWARE, *the stickler run*), she tried to be literal, making precise queries that included all the elements and satisfied all the conditions required in the topic specifications. In the second run (READWARE2, *the comprehensive run*), she incorporated the hits of the first run and added more hits using less precise queries. Here are two query examples for topic 420 (carbon monoxide poisoning):

Query 1

```
b: G:W P:5.0 poisoning H:2 S:1 S:2 !{"strychnine"}
```

Query 2

```
G:C P:2.0 !{"air quality"} !{chemical} !{emission S:37 T:367}
```

Query 1 is meant for the stickler run (labeled READWARE). We are looking for the word *poisoning* (G:W means word search) in the context of size 5 (P:5.0) of the probe H:2 (what are the numbers, implements the description phrase "*how widespread*") in documents from the baseline pool (b:) classified as having the subjects *Accidents/Crisis* (S:1) or *Crime/Police* (S:2) but not including the word *strychnine* anywhere. Document subjects were specified to fit the *poisoning* theme and to avoid getting ads or environmental documents that are excluded

by the narrative. The query returned 12 hit documents that were all accepted as relevant by the judges.

Query 2 is meant as an addition to the comprehensive run (labeled READWARE2) that includes all the hits from the stickler run. We are looking for the user topic *Carbon Monoxide* (T:367) using concept search (G:C) in a context of size 2 (P:2.0) in documents from the complete pool which are classified as having the subject *Medicine* (S:37, suggested by the word *symptoms* in the narrative) but not including the phrase “*air quality*” or the words *chemical* or *emission* anywhere in the document. The query returned 7 hit documents none of which were accepted as relevant by the judges (the analyst was right to exclude them from the stickler run).

A total of 12 queries were made for TREC topic 420. We retrieved only 38 hit documents for this topic in the stickler run. 27 were judged relevant. The total number of relevant documents found by the judges in 13 runs for topic 420 was 33. In the comprehensive run, we delivered 59 hit documents to the judges (including 38 from the first run) out of which only 28 hits were judged relevant. This consisted of 27 hits from the first run and only one hit from the comprehensive run.

4. Performance

The stickler run (labeled READWARE) showed very high selectivity and retrieved a total of only 3060 hit documents. Of these hits, 2019 were judged relevant. The stickler run had an average relevance rate of 66%. The average precision (non-interpolated) of the stickler run is 40% (3% better than last year). The R-precision (exact) is about 45% (1% better than last year).

The comprehensive run (labeled READWARE2) had a relevance rate of 48%. We retrieved only 5785 documents out of which 2774 were judged relevant. The average precision (non-interpolated) of this run is high at about 47% (10% better than last year). The R-precision (exact) is high at about 51% (7% better than last year). In this run (based on an evaluation over 13 runs), MITi scored best average precision in 23 topics compared to only 8 topics last year.

Figure 1 on the first page shows MITi performance this year in both runs as compared to our TREC-7 run of last year.

Figure 2 below graphs READWARE’s precision over X documents in both TREC 8 runs. This figure shows high precision in the first 30 documents retrieved in both runs.

5. Conclusion and Outlook

Knowledge managers and analysts both can enjoy direct access to highly relevant sets of documents retrieved by READWARE as predicted by the statistics in Figure 2 below and elsewhere. They can have high confidence that the text analysis performed by READWARE yields a larger pertinent measure of the “relevant whole” and that this reliably reduces the amount of information that needs to be read in further analyses. Any analyst would rather examine 3060 documents than 50,000 if given the choice, when they can be assured that the lower amount includes the most pertinent and relevant share of the available pool.

The TREC adhoc task is representative of the analytical tasks facing today’s knowledge managers, analysts and subject matter experts. Knowledge acquisition requires them to compare yesterday’s knowledge inventory (acquired perceptions and analysis

reports and the standards and decisions based on them) with today's knowledge input and calculate some measure of knowledge growth. They also need the means to identify current knowledge gaps and some notions of importance (what deserves attention? what is pertinent?). READWARE's knowledge types form a system of knowledge organization that is capable of meeting all these needs.

The TREC experiment shows that knowledge managers and analysts with or without subject matter expertise can use READWARE's knowledge types (concepts, topics, probes, issues and document sub-

jects) and the advanced search features (Boolean logic, sequence enforcement, context sizing, word search, phrase search and document-level search) to identify their knowledge gaps and formulate automatically executable information requests to fulfill their information needs (READWARE queries as formalized information requests).

READWARE is efficient at TREC text retrieval because it uses a coherent unifying framework of knowledge organization. This framework is layered, well-structured, expandable and even open to integrating custom models of knowledge organization.

READWARE® Document Precision at TREC 8

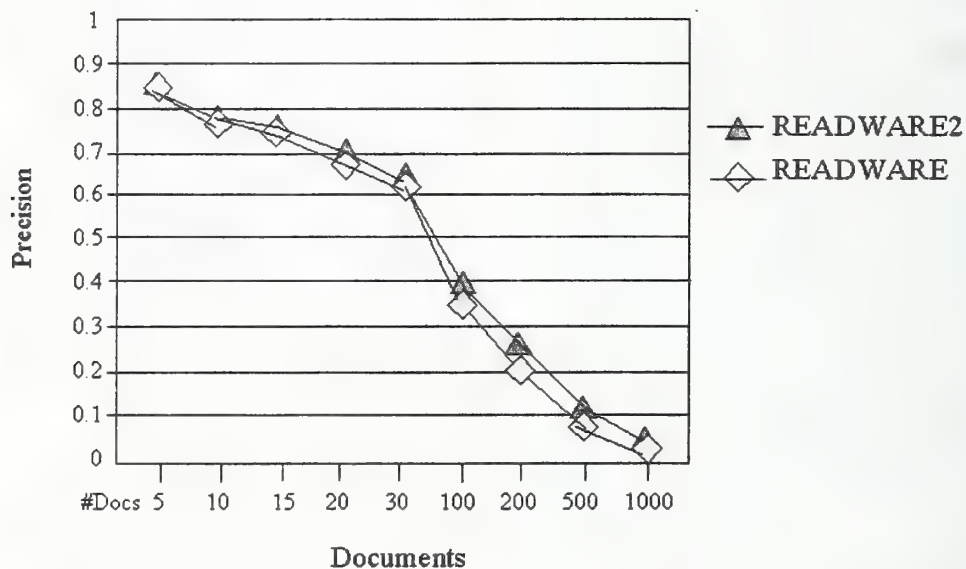


Figure 2.

A Sys Called Qanda

Eric Breck, John Burger, Lisa Ferro,
David House, Marc Light, Inderjeet Mani

The MITRE Corporation

{ebreck, john, lferro, dhouse, light, imani}@mitre.org

Introduction

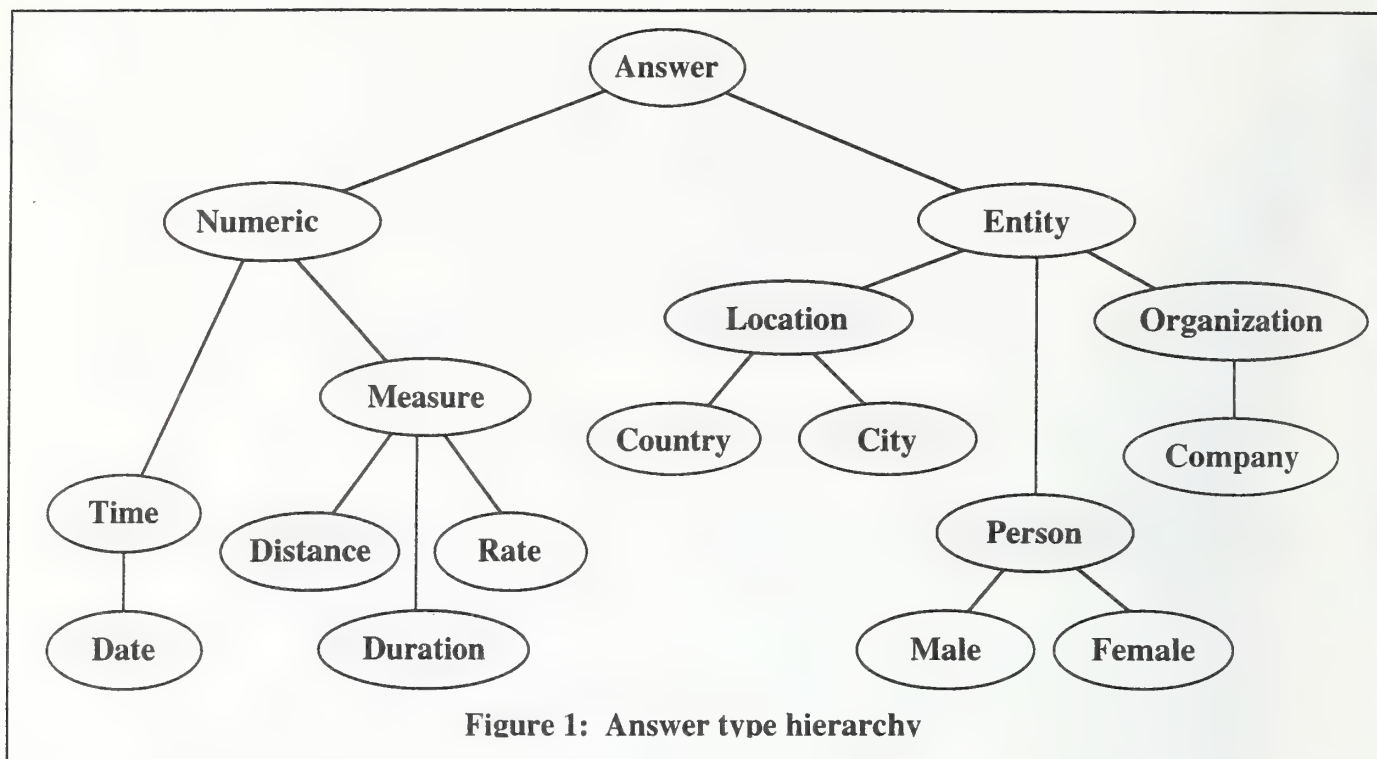
Our question answering system was built with a number of priorities in mind. First, we wanted to experiment with natural language processing (NLP) technologies such as shallow parsing, named entity tagging, and coreference chaining. We felt that the small number of terms in the questions coupled with the short length of the answers would make NLP technologies clearly beneficial, unlike previous experiments with NLP technologies on traditional IR tasks. At a more practical level, we were familiar with and interested in such technologies and thus their use would be relatively straightforward and enjoyable. Second, we wanted to use information retrieval (IR) techniques in hopes of achieving robustness and efficiency. It seemed obvious that many answers would appear in documents and passages laden with terms from the question. Finally, we wanted to experiment with different modules from different sites with differing input and output representation and implementational details. Thus, we needed a multi-process system with a flexible data format.

Driven by these priorities, we built Qanda,¹ a system that combines the finer-grained representations and inference abilities of NLP with IR's robustness and domain independence. In the following, we describe the Qanda system, discuss experimental results for the system, and finally discuss automating the scoring of question answering systems.

System Description

In broad strokes, Qanda processes a question as follows. It extracts the answer type from the question; e.g., PERSON is the answer type for *Who designed the Hancock Building in Boston?* At the same time, it hands the terms of the question to an IR search engine in order to retrieve relevant documents. Next it looks for elements in the retrieved documents that are of the right type; e.g., it might look for proper names of persons when answering a *who* question. Finally, it ranks these elements by comparing their contexts (i.e., sentences) with the question using term

¹ Qanda is pronounced *kwan•da*.



overlap. Thus, elements are returned which are of the right semantic type and whose contexts share some terms with the question. Based on all the references to an element and the corresponding contexts, an answer string is generated.

Let us look at each of these steps in more detail. The analysis of the question is performed using a set of specially designed patterns. These patterns are based on particular words and on part-of-speech tags. For example if the pattern **how (large|small|big)** is matched, the type MEASURE is returned. These answer types are arranged in the hierarchy shown in Figure 1 (where the types actually used by the rest of the system are shaded). Our question analyzer also has the ability to return types constructed in part from words in the question (e.g., MANNER_OF(DIE) is the type returned for *How did Socrates die?*) but we are currently not making use of these more open ended types.

An important component of Qanda is the IR engine that finds candidate documents. We have used the MG system, and experimented with Smart, but for the results described in this paper, we used the documents provided from the AT&T system. A parameter of the overall system is the number of relevant documents to examine for answers—we used the top ten.²

² Our intent was to process the top 100 documents returned by the IR component. After submitting our results, we discovered a trivial bug that limited subsequent processing to the top ten.

The relevant documents are then searched for elements of the right type. A special purpose tagger was designed for each of the answer types that could result from question analysis. Thus, we had taggers for person proper names, measure phrases, dates, organization names, etc. Many of the types are recognized by the Alembic system (Vilain and Day, 1996).

We should note that for both the processing done by these taggers and for the analysis of the question, a number of preprocessing modules are used. They include a punctuation tokenizer, a sentence tagger, and a part-of-speech tagger (Aberdeen et al., 1995).

Next the set of elements of the desired answer type are ranked. First they are ordered by how well they match the answer type. Since these types are arranged in a hierarchy it is possible to consider elements to be of the right type if they are of types above the desired type in the hierarchy. For example, a candidate LOCATION matches even if an answer type of COUNTRY is desired. However, a COUNTRY is preferred, i.e., candidates with types at least as specific as the answer type are preferred to those with a more general type. The elements of the right type, which we will call answer hypotheses, are ordered further by considering the textual contexts in which they occur. More specifically, we score each sentence that an element occurs in by how many terms it shares with the question. We do not currently weight these terms in any way; however, they are stemmed with the Porter stemmer and a small set of stop words is used. Answer hypotheses are ordered further by preferring hypotheses that occur earlier in their documents.

Until now, we have remained vague about what an element (or answer hypothesis) is. Qanda makes use of coreference and thus an element is not an offset in a document nor is it a string. It is more abstract: it corresponds to an entity that is discussed in the document. Coreference makes the ranking discussed above more complicated in that multiple contexts need to be considered for each element, i.e., every sentence in which the element is mentioned. Qanda uses the score for an element's best context as the score for the element.

Once the answer hypotheses have been ranked, answer strings are constructed for them. Qanda does this by combining the longest realization of an element with its best-scoring context. Consider the question about the Hancock Building given above. Further, consider the answer hypothesis corresponding to the individual I.M. Pei, mentioned in the following distinct sentences:

I.M. Pei is a well-known architect.
He designed the Hancock Building.
Pei studied at MIT.

For this answer hypothesis, Qanda constructs the following answer string: *I.M. Pei: He designed the Hancock Building.*

	250 byte	50 byte
Correct answer ranked 1	72	42
Correct answer within top 5	112	80
No correct answer	86	118
Mean RAR	0.434	0.281

Figure 2: Results on 198 TREC questions

If the above processing does not work, e.g., if no answer type is extracted from the question or no elements of the right type are found, Qanda uses the following fallback strategy: It looks for sentences in the relevant documents (from the IR search engine) that have a large term overlap with the question and returns these sentences as answer strings.

All of the modules mentioned above are glued together using a hub, which is a single executable that is configured to spawn a sub-process for each module and to set up a very simple file-based interprocess communication. XML is used for all data encoding. If the hub notices that a module is in an error state while processing a question it kills the module, restarts it, and goes on to the next question.

Finally, if for some reason no answer is found, as a last-ditch strategy, Qanda answers with the string *You're such a nice judge :-)*.

Discussion of Results

Our TREC results are given in Figure 2. Note that Qanda found the 250-byte answer and ranked it within its top five responses 56 percent of the time. It ranked a correct answer first 36 percent of the time. With respect to the 50-byte responses the percentages are 40 and 21, respectively.

Next we discuss a preliminary error analysis. The purpose of this was to guide the future development of Qanda. Errors were divided into the following categories, with overlap:

- **Question Analysis:** The question analyzer failed to recognize the question type and the actual question type was one of those in the hierarchy.
- **Preprocessing:** One of the above-mentioned preprocessors failed. For example, a sentence boundary was incorrectly marked.
- **Question-Answer Context Comparison:** The answer was found but not ranked highly enough. Additionally, a different method of comparing the answer context with the question would produce a higher rank.
- **Numeric Expressions (Date,Time,Measures):** An expression involving a number or referring to an element involving a number was misclassified.
- **Additional Answer Type Required:** The answer was of a type that is not part of our answer type hierarchy.
- **Coreference Resolution:** The extraction of the answer would have been facilitated by better coreference resolution of elements in the document.

- **Extra-Sentential Structure:** The extraction of the answer would have been facilitated by making use of rhetorical structure, discourse structure, document layout, etc. For example, the question *Why are electric cars less efficient in the north-east than in California?* would have been answered by our system if we would have recognized the relation between the sentence *John Williams ... points out that electric cars are less efficient in the northeast than they are in California* and the following sentence which reads *The cold in the north-east hurts our range.*
- **IR:** The answer was not contained in any of the top ten documents.
- **Bugs:** Errors that were caused by problems that were thought of and addressed, but incorrectly implemented.

Figure 3 shows these error categories and the number of questions for which Qanda could find no answer (again, there is some overlap, due to multiple errors). Note that over half of the errors can be attributed to the IR engine not ranking an answer document highly enough, in this case, in the top ten. As mentioned above, our intent was to process the top 100 documents, which would have almost certainly reduced the number of such errors. Also note that numeric expressions are surprisingly important in this test set (63 of the questions had numeric answers), and that we had a number of problems with such questions. Finally, our comparison of questions with potential answer contexts must clearly be made more sophisticated.

Question Analysis	2
Preprocessing	10
Question-Answer Context Comparison	18
Additional Answer Type Required	2
Numeric Expressions (Date,Time,Measures)	17
Coreference Resolution	9
Extra-Sentential Structure	3
IR	47
Bugs	7
Other	4
All errors	86

Figure 3: Error categorization

Preliminary results on automating the evaluation of question answering systems

Automated evaluations are crucial for tight system development loops, which in turn often result in greatly improved system performance. We explored word-based precision and recall as a

scoring metric for question answering systems. More precisely, we looked at scoring a system response to a question concentrating on the number of terms both in the response and in a human-prepared answer key for the question. The precision is the number of these overlap terms divided by the number of terms in the system response and the recall is the overlap divided by the number of terms in the answer key answer. The terms are stemmed and stop words are ignored. The answer key may contain multiple phrasings of a single answer and also multiple answers. The alternative in the answer key that provides the best F measure³ for a system response is used to generate the precision and recall numbers for that response.

We are interested in how well these metrics correlate to a judge's binary scoring of responses. We have scored all of the participating systems' responses against an answer key, constructed by our chief annotator⁴, based on her own knowledge, the responses, and the TREC corpus. Figures 4 and 5 plot a number of recall intervals on the horizontal axis and the number of systems responses that fall into the intervals on the vertical axis. The first graph is for responses deemed correct by the judges and the second for those responses deemed incorrect. We ignore precision since the system response length is roughly uniform. The results seem promising in that low recall correlates with incorrectness and high recall with correctness, with a coefficient of 0.84. However, we need to do a more careful statistical analysis and we need to explore the causes of the false positives and false negatives. A related study of answer key precision and recall metrics for automatic scoring of reading comprehension exams is described in (Hirschman, et al. 1999).

Conclusion

The mixture of NLP and IR that the Qanda system embodies has produced reasonable performance. Our error analysis indicates that performance can be increased by improving the modules that deal with numeric expressions and by improving the initial set of relevant documents considered. Simply considering more relevant documents for each question is likely to improve performance. In order to explore the impact of such parameter settings, we are using the automated evaluation method described above. We are also using this evaluation method to test configurations with and without certain modules (e.g., the coreference module). This will allow us to quantify the effects of that module.

³ F measure is the harmonic mean of precision and recall.

⁴ This individual has substantial experience in constructing language testing materials for adults but was not involved in the design or implementation of Qanda.

Acknowledgements

We would like to thank John Henderson, Warren Greiff, and Barry Schiffman for their help with the error analysis. We would also like to thank Lynette Hirschman for many helpful discussions.

References

- Aberdeen, John, John Burger, David Day, Lynette Hirschman, Patricia Robinson and Marc Vilain 1995. MITRE: Description of the Alembic System Used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference*. Morgan Kaufman.
- Hirschman, Lynette, Marc Light, Eric Breck, John D. Burger 1999. Deep Read: A Reading Comprehension System. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 325–332. College Park, Maryland.
- Vilain, Marc and David Day 1996. Finite-State Parsing by Rule Sequences. *International Conference on Computational Linguistics (COLING-96)*. Copenhagen, Denmark. The International Committee on Computational Linguistics.

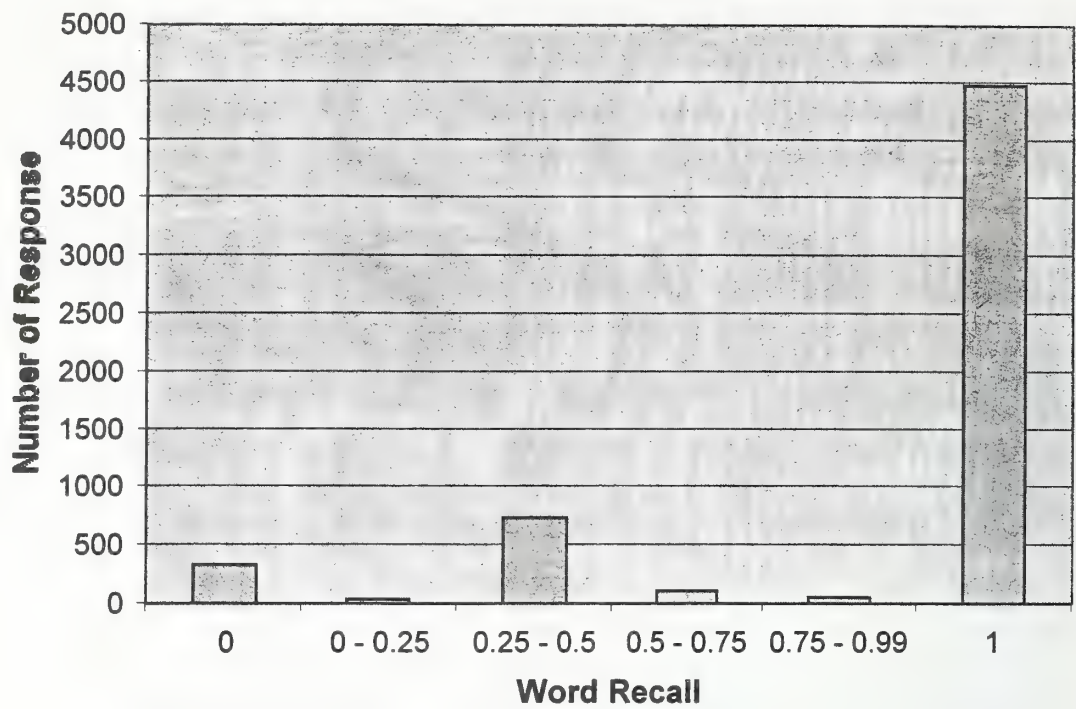


Figure 4: Automatic scoring for answers judged correct

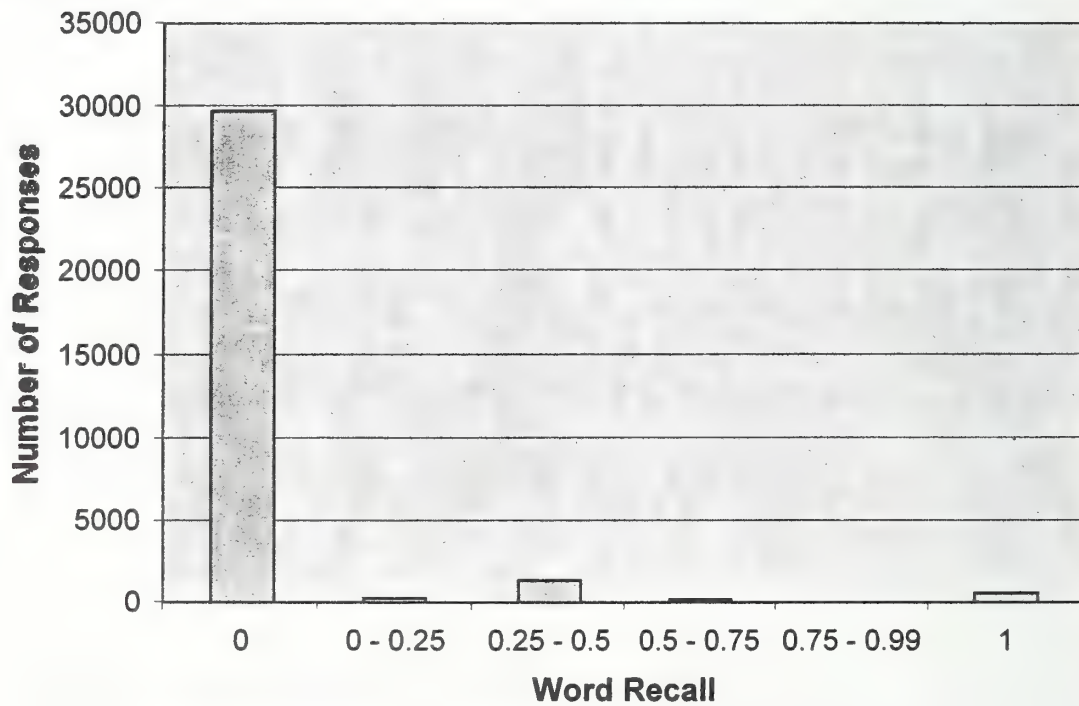


Figure 5: Automatic scoring for answers judged incorrect

Description of Preliminary Results to TREC-8 QA Task

Chuan-Jie Lin and Hsin-Hsi Chen

Department of Computer Science and Information Engineering

National Taiwan University

Taipei, TAIWAN, R.O.C.

E-mail: cjlin@nlg2.csie.ntu.edu.tw, hh_chen@csie.ntu.edu.tw

1. Introduction

Question Answering (QA) becomes a hot research topic in recent years due to the very large virtual database on the Internet. QA is defined to find the exact answer, which can meet the users' need more precisely, from a huge unstructured database. Traditional information retrieval systems cannot afford to resolve this problem. On the one hand, users have to find out the answers by themselves from the documents returned by IR systems. On the other hand, the answers may appear in any documents, even that the document is irrelevant to the question.

Two possible approaches, i.e., keyword matching and template extraction, can be considered. Keyword matching postulates that the answering text contains most of the keywords. In other words, it carries enough information relevant to the question. Using templates is some sort of information extraction. The contents of documents are represented as templates. To answer a question, a QA system has to select an appropriate template, then fill the template and finally offer the answer. The major difficulties in this approach are to find general domain templates, and to decide which template can be applied to answer the question.

Some other techniques are also useful. For example, to answer the questions "Who..." and "When...", the identification of named entities like person names and time/date expressions will help to locate the answer.

In our preliminary study, we adopt keyword-matching strategy coupling with expanding the keyword set selected from the question sentence by the synonyms and the morphological forms. We participate in the group "Sentence or under 250 bytes." The detail will be presented below.

2. Description of Our System

The system is composed of three major steps: (1) preprocessing the question sentences, (2) retrieving the documents containing answers, and (3) retrieving the sentences containing answers.

2.1 Preprocessing the Question Sentences

Our main strategy is keyword matching. This approach has a drawback, i.e., the words used in the question sentences and in the sentences containing the answers may be different. For example, verbs can be in different tenses and synonyms can also be used. Therefore we have to make necessary changes and expansions in the question sentences.

At first the parts-of-speech are assigned to the words in question sentences. Then, stop-words are removed. The remaining words are transformed into the canonical forms and selected as the keywords of the question sentences. For each keyword, we find all of its synonyms from WordNet 1.6. Those terms form an expansion set for the keyword. If the keyword is a noun, a verb, an adjective, or an adverb, all the possible morphological forms of the words in the expansion set are also added into this set. Here the morphological forms are the plural of a noun, different tenses of a verb, and the comparison of an adjective or an adverb. They are shown as follows:

noun AAA: AAAs | AA[s,z,sh]es

verb BBB: BBBed BBBing | BB[e]d BB[e]ing / BBBs | BB[s,z,sh]es

adjective or adverb CCC: CCCer CCCest | CC(y)ier CC(y)iest

The irregular nouns and verbs can be transformed by looking up the WordNet.

2.2 Retrieving the Documents Containing Answers

We implement a full text retrieval system to find the documents that may contain the answers. The purpose is to decrease the number of documents we have to search the answering sentences. Each keyword of a question sentence is assigned a weight, so are their various morphological forms. Those words tagged as NNP and NNPS, which denote proper nouns, have assigned higher weights. This is because they should be presented in the answer. The weights of added synonyms are less than the keywords. The score of a document is computed as follows:

$$score(D) = \sum_{t \in EX(T), t \text{ in } D} weight(T)$$

where T is one of the keywords, and $EX(T)$ its expansion set.

The document containing one keyword or any words in its expansion set earns a score of its weight. For example, consider the Question 30:

<num>Number: 30

What are the Valdez Principles?

Its keywords are “Valdez” and “Principles”, and the expansion sets are [valdez/valdezes/] [principle/principles/rule/rules/precept/precepts/rationale/rationales], respectively. If a document contains “principles” and “rules”, but no “valdez”, its score is determined by “principles” and “rules” only. The word “principles” gives a higher weight since it is proper noun, and the word “rules” gives a lower weight. The score of the document is the sum of these two weights.

Those documents that have scores no less than the threshold are selected as the answering documents. Threshold is set to the sum of weights of the words in the original question sentence. Note that the removed words have no scores. If no documents have scores greater than the threshold, we assume that no answers can be found for the question.

2.3 Retrieving the Sentences Containing Answers

Finally, we examine each sentence in the documents that may contain the answers. Those sentences that contain most words in the expanded question sentence are retrieved. The top five sentences are regarded as the answers. If there are more than five possible answers, we randomly select five of them. To meet the limit of 250 bytes, we truncate the sentences that exceed the limit. On the contrary, if the answer is shorter than the limit, we concatenate it with the next sentences.

3. Results and Discussions

The system run on the 198 questions provided by Q&A Track of TREC-8. The weights of proper noun keywords are set to 100, and the others are set to 1. Among these 198 questions, 60 have answers. Total 25 of them are correct, and 20 answers are at the top scores. The following shows some examples.

<num> Number: 29

What is the brightest star visible from Earth?

Ans: In the year 296036, Voyager 2 will make its closest approach to Sirius, the brightest star visible from Earth. Deep space is benign, so dust and cosmic rays will erode Voyager 2 extraordinarily slowly. In a billion or more years, Sagan said, "there w

<num> Number: 102

Who is the Voyager project manager?

Ans: Until December, Voyager 2 occasionally will glance at Neptune and dark space to improve the accuracy of observations its cameras and instruments made during the Neptune flyby, said Voyager project manager Norm Haynes. Pictures of empty space let engi

We examine the results of formal runs, and find that the system can be improved from several aspects:

(1) execution speed of the system

Owing to the long time required, 138 questions in the formal run do not have answers. After revising our algorithm and running again, we answer 136 questions. The evaluation is done by us ourselves. Total 62 of them are correct, and 42 answers are at the top scores.

(2) anaphor resolution

The answering sentence may contain pronouns or other anaphors referring the constituents in the previous sentences. We have to find the antecedents. Similarly, date expressions such as “today” have to be substituted by an exact time.

(3) phrasal searching

Phrasal searching is helpful in some kind of questions. For example, to answer the questions

<num> Number: 115

What is Head Start?

<num> Number: 40

Who won the Nobel Peace Prize in 1991?

the key phrases “head start” and “Nobel peace prize” are very useful to find the answers.

(4) question type

It is also helpful to identify possible answering candidates that the question is asking for. For example, the date/time expression is particularly preferred for the questions as “What day ...” or “When ...”. For questions asking about “How many people ...,” we shall offer a numerical answer.

Systems for name entity extraction in a famous message understanding competition (MUC, 1998) can be employed to provide this information.

(5) related words in different part-of-speech (POS)

We found that many answers are in different POSes from those in the question sentences. For example:

<num>Number: 130

When was Yemen reunified?

Ans: ... on 22 May 1990 , when the Yemeni community was reunified...

... the reunification of North and South Yemen in 1990...

Therefore, we have to add such related words to get better possibility to find the answer.

(6) time information

If the time is specific in the question, such as:

<num>Number: 32

Who received the Will Rogers Award in 1989?

we have to make sure if the answer contains information happening in the specific time. Time information can be mentioned earlier before the answering sentence, or mentioned in the header as the information of the whole document.

There are also cases requiring more semantic information or world knowledge to find the answers.

(1) additional knowledge

For example, the possessive expression “ ’s ” has many different meanings, depending on the relationship in the expression.

[Number: 7] What debts did Quintex group leave -- Quintex's debt

[Number: 11] President Cleveland's wife -- married 21-year-old Frances Folsom

[Number: 94] Who wrote the song, "Stardust"? -- Hoagie Carmichael's "Stardust,"

We have to know that Cleveland's wife is the one who married him, and Carmichael's "Stardust" means that he wrote this song. In this way, we can find the answer correctly.

Other examples are those phrases expressing the same information, but not using synonyms.

[Number: 14] producer of tungsten -- the biggest supplier of the metal

[Number: 34] Where is the actress, Marion Davies, buried? -- on her mausoleum

[Number: 108] created the Internet browser Mosaic -- Mosaic, developed by

We can see that the words “producer” and “supplier” are not synonyms, nor are “create” and “develop”. However, they do offer the same information.

(2) ellipsis

[Number: 104] ... in Marathi, the most commonly spoken local language ,

[Number: 111] The distance in time from Tokyo is ...

The answer to Question 104 in fact mentions “the most commonly spoken language in Bombay”, but the location is not shown. So is the answer to Question 111, which indeed mentions “from Tokyo to Niigata”.

4. Conclusion and Future Work

We propose a method to answer questions mainly based on keyword matching. The keywords in a question sentence is first selected, then all of their synonyms and morphological variants forms the expansion sets. Appropriate weights are assigned to each keyword.

To look for the sentences containing answers, we first employ a full-text retrieval system to select documents that may contain the answer. Then we examine each sentence in these documents to see if it offers the answer. Our approach answers 136 questions, 62 of them are correct, with 40 at top score.

In the future work, we will try to offer answers according to the types of questions. Besides, we will find the related words of keywords in different POSes, resolve anaphors, match patterns in phrasal level, and find the words missing in the sentences containing answers. Different scoring functions will be investigated to get better performance.

Reference

MUC (1998) *Proceedings of 7th Message Understanding Conference*, http://www.muc.saic.com/proceedings/proceedings_index.html.

CRL's TREC-8 Systems Cross-Lingual IR, and Q&A

Bill Ogden, Jim Cowie, Eugene Ludovik, Hugo Molina-Salgado,
Sergei Nirenburg, Nigel Sharples, Svetlana Sheremtyeva
Computing Research Laboratory, New Mexico State University
January 1999

Abstract

This paper describes the systems used by CRL in the Cross-lingual IR and Q&A tracks. The cross-language experiment was unique in that it was run interactively with a mono-lingual user simulating how a true cross-language system might be used. The methods used in the Q&A system are based on language processing technology developed at CRL for machine translation and information extraction.

Cross-Lingual IR

Can Monolingual Users Create Good Multilingual Queries?

Our interest in Interactive and Cross Language Text Retrieval has led to the design of a unique user interface for the cross language task. While many automatic techniques for query term translation and disambiguation have been proposed and tested, little work has involved the evaluation of a cross language system in combination with its user. We and others have proposed designing an interface that allows the user to help disambiguate terms provided by a system by providing "back-translations" of the system selected terms from which a monolingual user can select the appropriate meanings. The MULINEX system (<http://mulinex.dfki.de>) provides a query assistant feature with just such an interface. For our cross-language track experiment we wanted to see if these types of interfaces would help a monolingual user create good multilingual queries.

In our experiment, a single English speaker, who had little or no experience with German, French, or Italian, generated queries in each of these languages for the cross-language track run. For each topic, the user would read the English title, description, and narrative, and select the English terms from these sections judged to be the best query terms. They were only allowed to select terms that were contained in the original English topic. Then for each of the other target languages, the system showed extended English definitions of potentially relevant cross language query terms and phrases alongside their translations (see Figure 1). Only those terms that actually occur in the target data were presented to reduce the number of alternative terms. The user then selected the English definition that most accurately reflects the intention in the original query. The query terms selected for each language were used to retrieve and rank documents for that language and the results for all languages were merged into the final ranked list.

Retrieval

We conducted our cross-language runs using the Unicode Retrieval System Architecture (URSA) a multilingual retrieval engine that indexes and retrieves text using a common encoding scheme for all languages. Therefore, encoding for all texts were first converted to Unicode. URSA indexing of

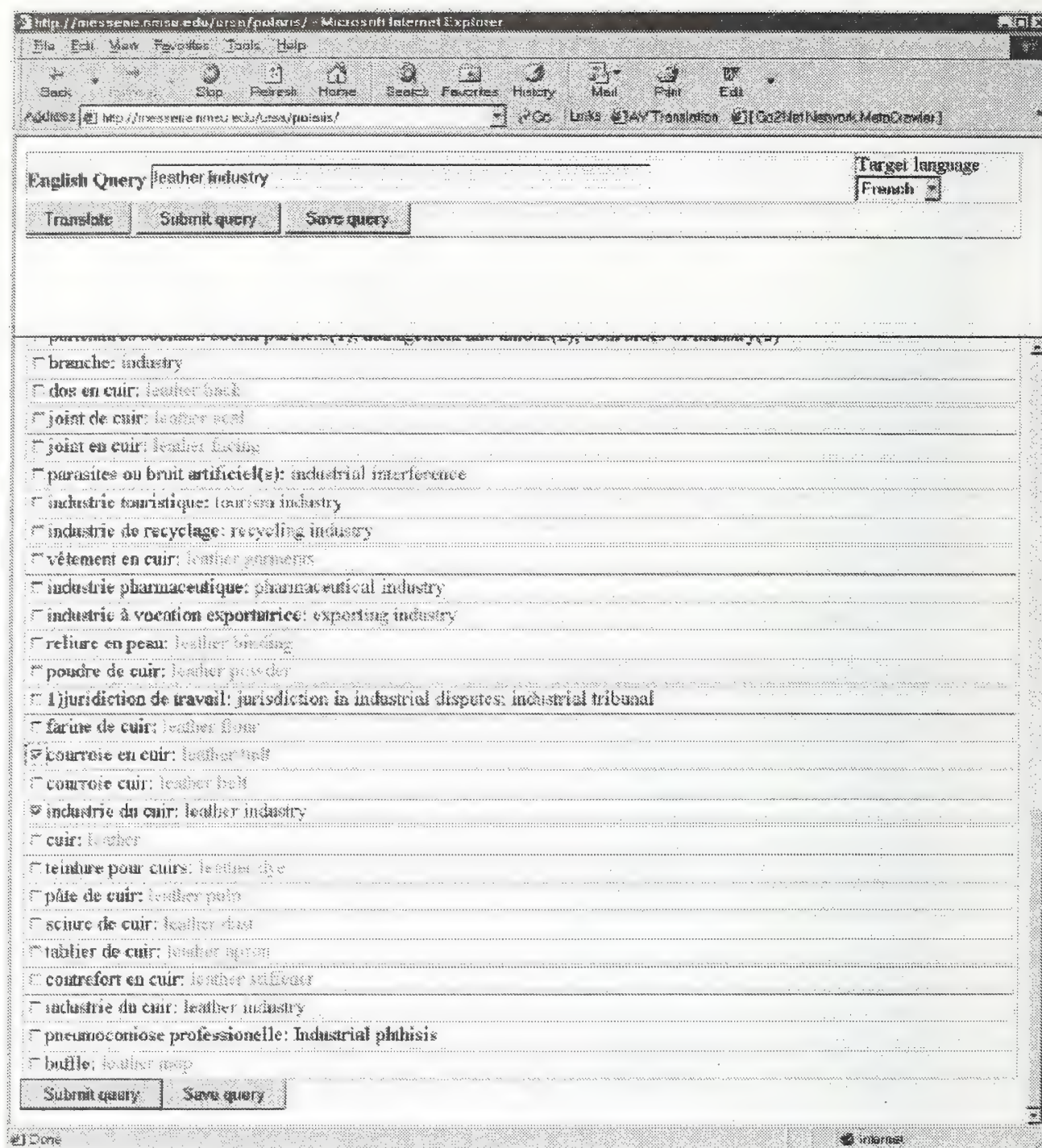


Figure 1. User-assisted cross-language query translation

the text used only simple stemming procedures specific for English, French, Italian and German. No other language specific compound word, phase indexing or other types of language processing was attempted. Consequently, one could expect an overall improvement in the performance of the base lined system, given the right effort. In this experiment we were only concerned with comparing the performance of the system when the cross-language queries were generated by the

system with help from a monolingual English-only speaker and the queries that were hand built by native language speakers and provided by NIST.

This was a preliminary experiment designed to test the feasibility of our approach. As usual the quality of the bilingual dictionaries will have a strong effect on the outcome. Some good query terms just were not present in the bilingual dictionaries used. In addition, our retrieval and ranking software could be better tuned to take advantage of the forms of the dictionary entries and phrases.

Merging

Merging the TREC multi-lingual queries is a constant issue for the cross-language studies. Our query system produces an ordered list (by score) of document ID's and the score for each language. The scores for each language are not comparable therefore the query results can not be merged using the score directly. Our technique was similar to that reported by the IBM group at TREC-7 [7] and involved obtaining a probability estimate that a returned document is relevant which and comparing these estimates between language retrieval systems. We used TREC-7 topics and results with our query system to obtain the performance for each language in terms of a sequence of relevance probabilities based on a precision score ordered by rank. To obtain the relevance probability we compared the results of the query system to the NIST supplied relevance tables (qrels) that specifies whether a document is relevant to a particular query. For each language, we generated a table mapping a rank index (from one to one thousand) to a precision score at that rank. For example, this tells us that a document at rank index 8 has a precision of 0.452, whereas a document at rank index 100 may have a precision of only 0.083. These rank-precision tables are roughly linear when plotted on a graph of precision vs. log (rank), so using linear regression an estimate of the relevance probability can be obtained for a given rank. These probability values are directly comparable between the different query systems, so the results for each query system can

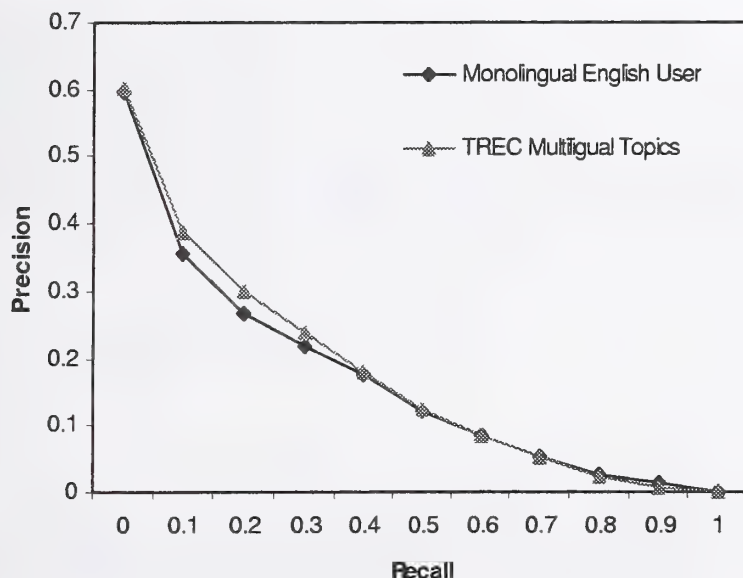


Figure 2. Precision –Recall performance of the cross-language retrieval system comparing the TREC supplied topics to those generated by the monolingual English user with help from the system

be merged using the probability value as a sort key.

Analysis

The primary analysis compares the results obtained by the monolingual user to the results obtained with the hand-translated queries provided by NIST for the cross language topics. As can be seen in Figure 2, the overall Precision/Recall curves for the two conditions are quite similar indicating that the user who knows no Italian, French, or German can use the system to generate queries that are as good as ones generated from the human translated TREC topics for these languages. The combined results shown in Figure 2 contain a large portion of English documents as well as the languages the user does not know. So, a more informative look at the data is shown in Table 1. Here the data show that indeed the English user is not doing as well as the baseline provided by the human translated TREC topics. For Italian and French, the user is doing about 85% of the baseline performance and for German it is worse (70 – 74 percent of the baseline).

	AveP	Recall At P(.20)
Italian – English User	.1770	.3249
Italian – Baseline TREC	.2077	.4027
% baseline	85%	81%
French - English User	.2722	.3980
French - Baseline TREC	.3236	.4682
% baseline	85%	85%
German – English User	.1110	.2297
German – Baseline TREC	.1592	.3103
% baseline	70%	74%

Table 1. Retrieval performance for individual languages comparing English user with TREC monolingual queries.

The fall off in performance Retrieval can have a number of reasons. For example the dictionaries that were used could have been lacking significant query terms or phrases. Indeed, if a query term could not be translated, the present system would not provide any alternative. Therefore, a number of simple improvements can be made to make the system better. With the more sophisticated tuning of the system, it can be expected that monolingual users will indeed be able to query in languages they cannot understand.

Question Answering

Extraction Based Method

CRL's approach to the Q&A problem is based on the Mikrokosmos Ontology [4]. The Ontology is intended to allow the representation of complex meanings. It consists of around 5,000 concepts linked using 200 relationship types. Each concept is linked to other concepts through up to 16 different relationships. The Ontology is being used principally to support machine translation, but recently we have been investigating its use as a control architecture for information extraction [2,3]. In this application a static template is defined by naming slots and defining potential slot fillers using the names of concepts from the Ontology. For example:

ELECTION

```
{ "ELECT", "ELECT" }  
{ "PERSON-ELECTED", "HUMAN" }  
{ "PLACE", "PLACE" }  
{ "DATE", "TIME" }  
{ "POSITION-ELECTEDTO", "SOCIAL-ROLE" }
```

defines an election template. The first element being the slot label, and the second the appropriate concept that must be attached to an element that would fill this slot. Our idea for question answering was to use the question to dynamically define such a template (partially filled with strings from the question), use a Boolean retrieval system to retrieve documents in which the key phrases, or equivalents occur, and extract the missing information -- the answer by carrying out the extraction process.

The amount of effort involved in this task was a total of six man weeks. Wherever possible off-the-shelf components were used. The Boolean retrieval was not completed in time for the evaluation, and the top five documents supplied by the AT&T retrieval engine were used. This had an impact on performance, as our whole method, at present, is dependent on information being localized in a single sentence in the document, which is not guaranteed with a general purpose ad-hoc retrieval.

Methodology

Our complete system consists of three main phases:

- Question Analysis - Recognize question structure and type
- Retrieval - Query building and document structuring
- Answer Generation - Sentence selection and answer selection.

Each of these is described briefly in the sections below.

Question Analysis

The basic processing undergone by the question and by sentences in the retrieved documents is the same. First the document is processed by a part of speech tagger, this marks each word in the sentence with one part of speech. In our current system we use a statistical tagger from MITRE. The text is run independently through the CRL Diderot name recognition system [5]. This recognizes names of organizations, places, people, and a variety of other units of interest (dates, money percentages etc.) The current complete list is shown in the table below. The labels are names of concepts from the Mikrokosmos Ontology.

Table of Elements recognized for the Q&A task

LINEAR-SIZE	ELECTRICITY	POPULATION-DENSITY	NATIONALITY
AREA	ENERGY	TEMPORAL-OBJECT	INHABITANT
VOLUME	VELOCITY	TIME-OBJECT	MATERIAL
LIQUID-VOLUME	ACCELERATION	AGE	EVENT-NAME
MASS	TEMPERATURE	NAME-HUMAN	PRODUCT-TYPE
RATE	COMPUTER-MEMORY	ORGANIZATION	NUMERIC-TYPE
PRESSURE		PLACE	DATE

The results of part of speech tagging and name and concept recognition are merged and the words are grouped into phrases, preference being given to the text units discovered by concept recognition. Verb and noun phrases and prepositional phrases are identified. A simple lexicon based stemming algorithm is then applied to the heads of all phrases and provides the citation forms needed to support lookup in the English to Ontology Lexicon.

Patterns are then applied to recognize noun phrase and verb phrase; phrases recognized by the name and measure recognition phase are not merged into noun phrases. In every case a head noun is identified. The head noun or verb is looked up in an English to Ontology lexicon. At this point we are ready to match the question against a set of skeletal question structures held in a "question lexicon". This allows the many ways that a question can be specified to all be mapped to a request for the same answer. Each entry consists of three parts:

<Type of Answer needed> <Additions to retrieval query> <Question pattern>

Where:

<Type of answer needed> specifies the ontological type of the answer needed

<Additions to retrieval query> specifies ontological concepts that should be mapped to lexical items to be used in the query process

<Question pattern> Is a pattern containing strings, which should be in the question, ontological types,, and Kleene stars, which allow matching any unit of question text. There is an implied "*" at the end of every question pattern. Currently there are some 500 question patterns in the system. Below we show the patterns used to handle questions on temperature.

Temperature Question Patterns

TEMPERATURE	THERMOMETRIC-UNIT	* what * temperature
TEMPERATURE	THERMOMETRIC-UNIT	* how hot
TEMPERATURE	THERMOMETRIC-UNIT	* how cold
TEMPERATURE	THERMOMETRIC-UNIT	* how many degrees
TEMPERATURE	THERMOMETRIC-UNIT	* how high * temperature
TEMPERATURE	THERMOMETRIC-UNIT	* how low * temperature

TEMPERATURE	THERMOMETRIC-UNIT	* what * melting point
TEMPERATURE	THERMOMETRIC-UNIT	* what * boiling point
TEMPERATURE	THERMOMETRIC-UNIT	* what * freezing point
TEMPERATURE	THERMOMETRIC-UNIT	* how many * THERMOMETRIC-UNIT

Temperature is a concept which is an object consisting of a NUMERIC-UNIT and a THERMOMETRIC_UNIT. The second element specifies that lexical entries attached to the concept THERMOMETRIC-UNIT should be included in the queries generated by the retrieval component of the system. The first pattern would recognize "At what temperature does tin melt?". The last pattern contains a concept in addition to strings, in lower case. This would match questions such as "How many degrees centigrade is the melting point of tin?".

The question recognition system uses dynamic programming to select the closest matching question pattern. Strings are matched with strings in the question, and concepts are matched with the head concepts found for each phrase. If a direct match is not found the concept's parent in the "IS-A" hierarchy will also be tried. This information is then passed both to the retrieval system query builder and to the answer extraction system.

Retrieval

The query building component of the system was not integrated in time for use in this evaluation [1]. Instead the top 5 documents returned by the AT&T system, which were provided for the evaluation, were used. A brief description of the eventual operation of the query builder is given here.

Our goal is to find a text with a single sentence which specifies the answer in the context of all the constraints of the question. However, the constraints may need to be relaxed, and synonyms generated to allow a matching sentence to be found. The query system also expands the answer indicator concepts using the ontological lexicon. The THERMOMETRIC-UNIT will become "centigrade OR fahrenheit OR kelvin OR c OR f OR k". A boolean retrieval system is used and the initial query attempts to find all the phrases in a single sentence. If this fails then a second retrieval is attempted using head words. A third retrieval is attempted where head words are substituted by their synonyms. If all the above fail then the synonym query is retried with the constraint that all the terms are in a paragraph.

The benefits of giving all the terms in a question equal weighting, and of only performing stemming and term expansion in response to the initial query failure, are that texts are obtained where all the information specified is found in a close context.

Document Structuring

The retrieved documents undergo the same language processing steps as was carried out on the query. Each sentence is part of speech tagged. Name recognition is run on whole documents, which allows much more accurate performance than processing single sentences. Phrases are recognized, and heads of phrases are looked up in the English to Ontology lexicon. The resulting structure, for each document, is then passed to the question answering phase.

Answer Generation

The structured question is used as a template and matched against each sentence in the document. Each sentence receives a score for each string and each concept in the question which matches a text unit in the sentence. If no text unit matches the concept required then the sentence is rejected, otherwise the answer string is produced accompanied by a score for the number of question slots filled in producing this answer. A high number of slots gives a high score. Once all the documents have been processed all the answers are sorted by score and the top five picked. In this preliminary system the answer selection process only requires the answer concept and does not specifically check that the expected answer object is present. Thus *tall* would be an acceptable answer for a linear size question. For the TREC tasks the answers were expanded on either side up to the maximum allowable number of bytes containing whole words. The initial answers produced by

Sample Question and Answer

The following shows a question, the resulting structure, and the set of answers obtained, all from the same document from the LA Times.

% How tall is the Eiffel Tower?

answer-indicator LINEAR-UNIT
np "the Eiffel Tower" "tower"

LA061789-0071	2.0 CRL 1,000-foot
LA061789-0071	2.0 CRL short
LA061789-0071	0.95 CRL 76-foot
LA061789-0071	0.95 CRL 90-foot
LA061789-0071	0.95 CRL Too Tall

Performance

Results were submitted for the 250 byte and the 50 byte tasks.

250 Byte Responses

33 - no answer
89 - no correct answer
78 - correct answer in 5 responses
Mean rank - 0.268

50 Byte Responses

33 - no answer
97 - no correct answer
70 - correct answer in 5 responses
Mean rank - 0.22

Future Work

Our top priority at the moment is to get retrieval integrated into the system. More sophisticated algorithms for answer selection are also required. For example not producing as an answer something which was specified in the question; some modicum of syntax will also help in matching

sentences to the question template. The question lexicon needs to be expanded to cover more question types.

A web search version will be built to allow the demonstration of the process on non-TREC data. Other knowledge sources will be incorporated to handle answers that are unlikely to be explicitly specified in documents (What is the capital of France?). The method is not language independent, but the components used part of speech tagging, phrase recognition, name recognition and an ontological lexicon are already available for Spanish and Chinese, so the development of question answering systems for these languages should be possible in a relatively short period of time [6,8].

References

- [1] Cowie, J. 1999, Collage: An NLP Toolset to Support Boolean Retrieval, in "Natural Language Information Retrieval" editor: Tomek Strzalkowski, Kluwer Academic Publishers
- [2] Cowie, J. and Y. Wilks 1999 *Information Extraction*, in "A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text", Editors: Robert Dale, Hermann Moisl, and Harold Somers. Marcel Dekker Inc 1999
- [3] Cowie, J., E.Ludovik, H. Molina-Salgado. 1998. *Improving Robust Domain Independent Summarization*, proceedings of "Natural Language Processing and Industrial Applications", Moncton, Canada, 1998
- [4] Mahesh, K., and S. Nirenburg, 1995, A situated ontology for practical NLP, in Proceedings of the workshop on Basic Ontological Issues in Knowledge Sharing, International Joint Conference on Artificial Intelligences (IJCAI-95), Montreal, Canada.
- [5] Cowie, J., Guthrie, L., Wakao, T., Jin, W., Pustejovsky, J., and Waterman, S. 1993. The Diderot Information Extraction System. In Proceedings of the First Conference of the Pacific Association for Computational Linguistics, (PACLING 93), Vancouver, Canada
- [6] Cowie, J. 1996. CRLs approach to MET (Multilingual Named Entity Recognition), Proceedings of the Tipster Text II 24 Month Workshop, Morgan Kaufman, May 1996
- [7] M. Franz, J. McCarley, S. Roukos, "Ad hoc and Multilingual Information Retrieval at IBM" *Proceedings of the Seventh Text Retrieval Conference (Trec-7)*. E.M. Voorhees and D.K. Harmon, Editors, NIST Special Publication, 500-242. 1998.
- [8] Sheremetyeva S., J. Cowie, S. Nirenburg and R. Zajac. *A Multilingual Onomasticon as a Multipurpose NLP Resource*. 1998. Proceedings of the "First International Conference on Language Resources and Evaluation", Granada, Spain.

NTT DATA: Overview of system approach at TREC-8 ad-hoc and question answering

Toru Takaki

Research and development Headquarters

NTT Data Corporation

Kayabacyo Tower Bldg., 1-21-2, Shinkawa,

Chuo-ku, Tokyo 104-0033 Japan

E-mail: takaki@rd.nttdata.co.jp

1 Introduction

In TREC-8, NTT Data Corporation participated in the ad-hoc task and question answering track. In this paper, we describe our system approach and discuss the results. The summary of each task of our approach is shown below.

Ad-hoc

We submitted five results as official runs. Two kinds of results, long query results (title, description and narrative) and short query results (title and description), were submitted. Another kind of result that applied query expansion technique or not applied was also submitted. In our work at TREC-8, we concentrated our interest on extraction of the query terms. Specifically, we applied a removal technique of negative information in topics and specification of multiword phrases.

Question Answering

The question answering (QA) track is first attempt in TREC. We gave priority to the following verification for the QA track: (1) the effectiveness of technique by surface-text-based information in the text and (2) application of the information extraction technique. In our QA track, the following processing was done: (1) decision of answer form by question analysis, (2) passage scoring and selection for detailed analysis of the answer after initial retrieval, and (3) information extraction that look for words or phrases that match the form of the answer. We submitted two results to the answer categories of different strength respectively. A retrieval technique like ad-hoc is effective in a category answered by 250 bytes or less in our evaluation but the question analysis is important for a stricter category answered by 50 bytes or less.

2 Ad-hoc task

We concentrated our interest on the query term selection from the topics. The terms describing criteria of non-relevance in the topic sentence were not applied as query terms. Multiword phrases and each term

composing them are applied as query terms. And, pseudo relevance feedback was done in the query term expansion. This method is similar to Local Context Analysis (LCA) [3].

2.1 Approach

In ad-hoc, we processed the retrieval as follows.

Index

The index was made from stemmed text within <TEXT> and </TEXT> tags in the data set of TREC-8 (in TREC disks 4 and 5: the *Financial Times* 1991-1994, *Federal Register*-1994, *Foreign Broadcast Information Service* and the *LA Times*). In our last TREC-7, four indexes by document source were made, and the relevance ranking processing was done for each index, then those results were merged into one result [1]. However one index was constructed, and the retrieval processing was done in TREC-8.

Search and Relevance ranking

Our ranking processing has 4 steps:

Step 1: Query term selection from topics

- (1) Deletion of negative sentences from topics
Sentences discussing criteria of non-relevance in the narratives (such as "Documents that describe... are not relevant.") are removed.
- (2) Deletion of stopwords and stemming
The stopwords were deleted by using the list of 550 terms. Moreover, stemming was applied to the terms within the topics.
- (3) Extraction of multiword phrases
The multiword phrases were extracted by using a part-of-speech tagger. This procedure was applied to only the title part of topics for all submitted results. Moreover, only two word phrases were extracted and not applied this procedure to the multiword phrases more than three words. Terms other than the multiword phrase were also extracted as query terms.
- (4) Weighting for query terms
The word that composed the multiword phrase, were used as query terms. Moreover, each query term was given a weight that decided by which topic category in or whether multiword phrase.

Step 2: Initial retrieval

In TREC-8, we did the relevance ranking by using the BM25 function of Okapi [2]. The function is shown as follows.

$$\sum_{T \in Q} w^{(1)} \frac{(k_1 + 1)tf}{K + tf} \frac{(k_3 + 1)qtf}{k_3 + qtf} \quad (1)$$

where Q is a query, containing terms T ,
 $w^{(1)}$ is the Robertson/Sparch Jones weight of T in Q ,

$$w^{(1)} = \frac{(r + 0.5)/(R - r + 0.5)}{(n - r + 0.5)/(N - n - R + r + 0.5)} \quad (2)$$

N is the number of documents in the collection,
 n is the number of documents containing the term,
 R is the number of documents known to be relevant to a specific topic,
 r is the number of relevant documents containing the term,
 K is $k_1((1 - b) + bdl/avdl)$,
 tf is the frequency of occurrence of the term within a specific topic,
 qtf is the frequency of the term within the topic from Q was derived, and
 dl and $avdl$ are the document length and average document length.

First, the query terms, which selected with step 1, was input to the system with query word weight, and the initial retrieval result was obtained. The results were submitted before the query expansion (**nttd8al,nttd8am**).

Step 3: Query term expansion

A method similar to LCA [3] was adopted as a query term expansion technique. A passage importance score is given to each passage unit and extended terms are selected in LCA. Since our implementation of LCA is not complete, the top n ranked documents of the initial retrieval were used instead of the passage. The data set used for query expansion is the same Disks 4-5 data set as was used for the retrieval data.

Step 4: The second retrieval processing

This retrieval processing was the same as that in step 1 was used. Query terms that were extracted by query expansion were added to the original query terms. In this case, the weights of the query terms are given to the expanded query terms.

2.2 Result and analysis

We submitted five results. Three are by long query and two are by short query. The same ranking method and parameters were used regardless of the retrieval type (long or short). In the three long query results, **nttd8le** is query expanded, **nttd8l** has no query expansion and **nttd8lx** is a hybrid of **nttd8l** and **nttd8le**. In the two short query results, **nttd8me** is query expanded and **nttd8m** has no query expansion. The parameters used for the TREC-8 experiments were as follows. For the BM25 function, $k_1=1.0$, $b=0.5$ and $k_3=0$. The weight of the extracted multiword phrase was 1.5, each word in the extracted multiword phrase was 0.8 and weight of the other words was 1.0. The retrieval result is shown in Table 1.

Run	AveP	Rel _{ret}	#Q \geq med	P10	P30
nttd8al No expansion (T+D+N)	0.2781	2973	36	0.4880	0.3800
nttd8ale Expanded (T+D+N)	0.2921	3120	40	0.4940	0.3847
nttd8alx hybrid of nttd8al and nttd8ale	0.2817	2986	38	0.4760	0.3840
nttd8am No expansion (T+D)	0.2649	2937	39	0.4600	0.3667
nttd8ame Expanded (T+D)	0.2721	3028	39	0.4900	0.3747

Table 1: Submitted ad-hoc retrieval runs

used topics	AveP		
	Basic query processing (baseline)	Removal of negative information(from N)	Extraction of multiword phrase (from T,D and N)
T	0.2322	No change	0.2386
D	0.2386	No change	0.2322
T+D	0.2714	No change	0.2714
T+D+N	0.2731	0.2820	0.2820

Table 2: Ad-hoc retrieval runs for various processing types

Ad-hoc basic processing

In the basic retrieval processing, we analyzed the results by used part of the topics. The results show that retrieval becomes better as queries get longer (Table 2).

Removal of non-relevant topic sentences

The result of removing a negative sentence and that of the basic retrieval processing are shown in Table 2. This processing, removing the negative, is done only in the narrative part, so the results do not change in the basic retrieval processing, which does not use the narrative part. This processing results in a 3.3% improvement in average precision.

Phrase identification

Table 2 shows the results for multiword phrase processing of two words in all the topic parts. The result did not change too much, although it was successful in the topics of TREC-7.

3 Question Answering Track

This section describes our method adopted for the question answering track and discusses our results.

3.1 Approach

In our QA track, processing was executed according to the following steps:

- (1) Decision of answer forms by question analysis,
- (2) Selection of candidate documents and parts for detailed analysis by an initial search of the documents, and
- (3) Information extraction to output the final results from the candidate parts.

We mainly used adopted method that depended on surface-text-based.

Decision of answer forms by question analysis

Step 1:

Specifies the part of speech in the question by the POS tagger.

Step 2:

The answer forms of each question were decided according to wh-determiner, wh-pronoun, wh-adverb, etc. The correspondence of the part of speech and the answer form was manually made as a table. The number of answer forms was 24. For instance, when the question is "How long ?" and the answer form is assumed to be "TIME". For Question 127 "Which city has the oldest relationship as a sister-city with Los Angeles?", three answer forms are sequentially given. The prime candidate of the answer form is "CITY", the second is "LOCATION", and the third is "PROPER". Here, when the answer form was not able to be specified, an answer form are given as "UNKNOWN", and the subsequent information extraction is not performed.

Initial search

Step 1: Execution of initial search

Our System is based on the BM25 algorithm. The initial search is the same as the one used in our ad-hoc. The query terms were extracted from the question. After the initial search, the top n ranked documents (D_1, D_2, \dots, D_n) were to be answer extraction candidate documents (n was assumed to be an evaluation parameter). Some passage parts where the appearance density of the query term was high was extracted from the top n ranked documents of the initial search. These parts were assumed to include an answer. Moreover, a score was given to the extracted answer candidate part. The method of scoring is as follows.

Step 2: Scoring the answer candidate part

The score $s(P_{ij}, Q_k)$ for query term Q_k and each term position P_{ij} in document D_i is given (P_{ij} is the j -th term position from the top of document D_i). When query term Q_k appears at P_{ij} , the IDF measure of Q_k is given to P_{ij} , that is, $s(P_{ij}, Q_k) = IDF(Q_k)$. The score at the circumference term position was related to the distance with term position appearing query term Q_k . The distance with appearing query term Q_k reduces the score (Figure 1). Two kinds of scoring method were executed.

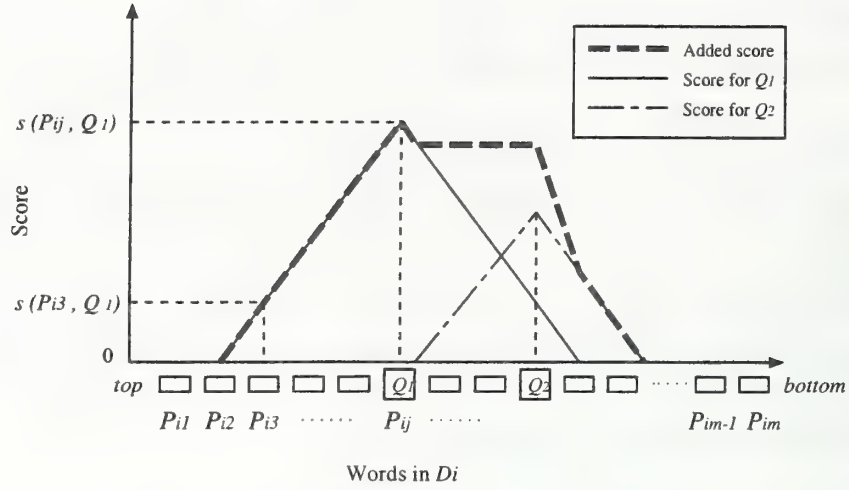


Figure 1: Method of giving term position score.

Methods of giving the term position score

Method X1: The scores are given to the term positions within a fixed number of terms from P_{ij} .

Method X2: The scores are given to the term positions within the range corresponding to the IDF value of the query term Q_k .

Final score $s'(P_{ij})$ at term position P_{ij} was finally assumed to be the sum total of the $s(P_{ij}, Q_k)$ given to each query term. That is, a higher score was given to the part where the appearance density of the query term was high. The consecutive passage parts where the score $s'(P_{ij})$ was more than a set threshold were decided the answer candidate parts. Here, the answer candidate parts are assumed to be C_{ip} . Maximum score $s'(P_{ij})$ in C_p was assumed $sc(C_{ip})$ which the score of the answer candidate parts, that is, $sc(C_{ip}) = \max(s'(P_{ij}))$.

Next, the answer text that suits the answer form specified by the query demand is found from the answer candidate parts. When a candidate part C_{ip} included text that matched the answer form, $sc(C_{ip})$ was added as a bonus score.

Information extraction of answer form text

In this approach, information types were given to the each text by the information extraction with the rule-based. Moreover, the name of a country and the city name, etc. was used for information extraction as dictionary information (Table 3), but we did not use corporate name's dictionary and etc., which was not able to be prepared. The number of last-name entries was 88798 but only 1000 general names was used. When an information type that suits the answer type given by the query demand appears, the scores of the candidate part are added. In this case, two kinds of score adding methods were adopted.

Methods of adding score by answer type

Method Y1: The score is added without considering where information that fits the answer type appears.

Dictionary	Number of Entries	Data Source	Examples
Countries	253	ISO 3166 codes	Japan, USA
Cities (airport cities)	1140	www.ufreight.com	Los Angeles, Tokyo
World regions	14	www.yahoo.com	Oceania, Europe, Arctic
US states	50	www.yahoo.com	Maryland, Kentucky, Illinois
Currency names	221	www.bloomberg.com	Euro, European Currency Unit, French Franc
Currency abbreviations	164	www.bloomberg.com	USD, JPY
Dates and times	54	Hand entered	Sunday, Apr, a.m.
Last name	1000	www.census.gov	Smith, Johnson, Williams, Jones

Table 3: Dictionary Features

Runs	Answer length	n (Num. of used initial top ranked documents)	Scoring method X	Scoring method Y
nttd8qs1	50	10	X2	Y2
nttd8qs2	50	10	X2	Y1
nttd8ql1	250	10	X2	Y2
nttd8ql4	250	30	X1(25 words)	Y1

Table 4: Parameters used for the submitted QA runs

Method Y2: The addition degree of the score is proportional to the distance between the term position where the score in answer candidate text C_{ip} is the maximum and the term position of the information that fits the answer type.

Text within a specified number of bytes (50 or 250) is extracted from the peripheral part of text suitable for the answer type, and the final answers are outputted. Our TREC-8's goal was to extract text around the answer as well as the answer.

3.2 Result and analysis

We submitted two results to the category of "Under 50 bytes" and "Sentence or under 250 bytes". **nttd8qs1** and **nttd8qs2** are for the under-50-bytes category, and **nttd8ql1** and **nttd8ql4** are for the under-250-bytes category. The parameters used for each run is shown in Table 4. The result is shown in Table 5. Our result was better than the average of all participants. However, there were a lot of questions in which the correct answer was not able to be included in the top five outputs. Table 6 is the result of classifying by the question to be given the answer form except "UNKNOWN" and to be given "UNKNOWN". Our mean reciprocal rank was much lower in under-50-bytes category when the answer form was not able to be specified. In contrast, the rank was high when it was possible to specify the answer form. However, this difference in rank did not appear in the under-250-bytes category. Therefore, we think a retrieval technique like ad-hoc is effective for under-250-bytes category in our evaluation, and

Run	Mean_ Reciprocal_rank	Num. of answers found at rank X						#Q	#Q
		1st	2nd	3rd	4th	5th	Not found	Best	≥ Med
nttd8qs1	0.273	40	17	9	6	5	121	54	168
nttd8qs2	0.259	37	13	14	7	7	120	49	160
nttd8ql1	0.439	65	32	9	7	6	79	75	183
nttd8ql4	0.371	54	25	10	8	8	93	62	182

Table 5: Submitted QA runs

Classification	#Question	Mean_reciprocal_rank			
		50 bytes		250 bytes	
		nttdqs1	average	nttdql1	average
UNKNOWN	48	0.077	0.229	0.520	0.338
Expect UNKNOWN	150	0.335	0.209	0.413	0.330
All	198	0.273	0.214	0.439	0.332

Table 6: Classified analysis by answer form

that the question analysis is important for the stricter under-50-bytes category.

4 Summary

We described our system approach and discussed the results for ad-hoc and question answering in TREC-8. Especially, our results in question answering track were a little fine. Our implementation is not complete with respect to the answer form specific processing and the information extraction processing, so there are a lot of points that should be improved. Moreover, we will examine linguistic techniques for question answering in the future.

References

- [1] H. Nakajima, T. Takaki, T. Hirao and A. Kitauchi. NTT DATA at TREC-7: system approach for ad-hoc and filtering. In E. M. Voorhees and D. K. Harman editors, *Proceedings of the 7th Text Retrieval Conference (TREC-7)*, pages 481-489. NIST Special Publication 500-242, 1999.
- [2] S.E. Robertson, S. Walker and M. Beaulieu. Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive. In E. M. Voorhees and D. K. Harman editors, *Proceedings of the 7th Text Retrieval Conference (TREC-7)*, pages 253-264. NIST Special Publication 500-242, 1999.
- [3] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*, pages 4-11, Zurich, 1996.

Do Batch and User Evaluations Give the Same Results?

An Analysis from the TREC-8 Interactive Track

William Hersh, Andrew, Turpin, Susan Price, Dale Kraemer,
Benjamin Chan, Lynetta Sacherek, Daniel Olson
Division of Medical Informatics & Outcomes Research
Oregon Health Sciences University
Portland, OR, USA

An unanswered question in information retrieval research is whether improvements in system performance demonstrated by batch evaluations confer the same benefit for real users. We used the TREC-8 Interactive Track to investigate this question. After identifying a weighting scheme that gave maximum improvement over the baseline, we used it with real users searching on an instance recall task. Our results showed no improvement; although there was overall average improvement comparable to the batch results, it was not statistically significant and due to the effect of just one out of the six queries. Further analysis with more queries is necessary to resolve this question.

Introduction

A great deal of information retrieval (IR) evaluation research dating back to the Cranfield studies [1] and continuing through the Text Retrieval Conference (TREC) [2] is based on entering fixed query statements from a test collection into an IR system in batch mode with measurement of recall and precision of the output. It is assumed that this is an effective and realistic approach to determining the system's performance [3]. Some have argued against this view, maintaining that the real world of searching is more complex than can be captured with such studies. They point out that relevance is not a fixed notion [4], interaction is the key element of successful retrieval system use [5], and relevance-based measures do not capture user performance in some domains such as medicine [6].

The TREC Interactive Track is designed to assess real-user searching in the system-oriented TREC evaluation milieu. For the past three years (TREC-6, TREC-7, and TREC-8), the track has employed an "instance recall" task, where users are asked to identify instances of a topic [7]. Instance recall is defined as the fraction of total instances (as determined by the NIST assessor) for the topic that are covered by the documents saved by the user. Also measured is instance precision, which is the fraction of saved documents that contain one or more instances. Figure 1 shows two example queries from this year's track. The track allows a variety of hypotheses about IR systems to be evaluated with real users.

The goal of our effort for this year's Interactive Track was to assess whether IR approaches achieving better performance in the batch environment could translate that effectiveness to real users. This was done by first transforming queries, documents, and relevance judgements from the TREC-6 and TREC-7 interactive tracks into a test collection that could identify highly effective batch performance compared to a baseline. In particular, we focused on the newer weighting schemes that have shown to be effective with TREC data over the standard TF*IDF baseline. The most effective approach was chosen to serve as the "experimental" system while the standard TF*IDF served as the "control" system. Since we compared weighting schemes - a back-end functionality - the user interface for both systems was identical. We also evaluated two different searcher populations - librarians (mostly non-medical) and graduate students.

Number:
414i
Title:
Cuba, sugar, imports
Description:
What countries import Cuban sugar?
Instances:
In the time allotted, please find as many DIFFERENT countries of the sort described above as you can. Please save at least one document for EACH such DIFFERENT country.
If one document discusses several such countries, then you need not save other documents that repeat those, since your goal is to identify as many DIFFERENT countries of the sort described above as possible.

Number:
428i
Title:
declining birth rates
Description:
What countries other than the US and China have or have had a declining birth rate?
Instances:
In the time allotted, please find as many DIFFERENT countries of the sort described above as you can. Please save at least one document for EACH such DIFFERENT country.
If one document discusses several such countries, then you need not save other documents that repeat those, since your goal is to identify as many DIFFERENT countries of the sort described above as possible.

Figure 1 - Sample queries from the TREC interactive track.

Experiment 1 - Finding an effective weighting scheme for experimental system

The goal for the first experiment was to find the most effective batch-mode weighting scheme for interactive track data that would subsequently be used in interactive experiments. All of our batch and user experiments used the MG retrieval system [8]. MG allows queries to be entered in either Boolean or ranked mode. If ranking is chosen, the ranking scheme can be varied according to the Q-expression notation introduced by Zobel and Moffat [9].

A Q-expression consists of eight letters written in three groups, each group separated by hyphens. For example, BB-ACB-BCA, is a valid Q-expression. The two triples describe how terms should contribute to the weight of a document and the weight of a query respectively. The first two letters define how a single term contributes to the document/query weight. The final letter of each triple describes the document/query length normalization scheme. The second character of the Q-expression details how term frequency should be treated in both the document and query weight, e.g., as inverse document/query frequencies. Finally, the first character determines how the four quantities (document term weight, query term weight, document normalization, and query normalization) are combined to give a similarity measure between any given document and query. To determine the exact meaning of each character, the five tables appearing in the Zobel and Moffat paper must be consulted [9]. Each character provides an index into the appropriate table for the character in that position.

Although the Q-expressions permit thousands of possible permutations to be expressed, several generalizations can be made. Q-expressions starting with a B use the cosine measure for combining weights, while those starting with an A do not divide the similarity measure through by document or query normalization factors. A B in the second position indicates that the natural logarithm of one plus the number of documents divided by term frequency is used as a term's weight, while a D in this position indicates that the natural logarithm of one plus the maximum term frequency divided by term frequency is used. A C in the fourth position indicates a cosine measure based term frequency treatment, while an F in this position indicates Okapi-style usage [10]. Varying the fifth character alters the document length normalization scheme. Letters greater than H use pivoted normalization [11].

Methods

In order to determine the best batch-mode weighting scheme, we needed to convert the prior interactive data (from TREC-6 and TREC-7) into a test collection for batch-mode studies. This was done by using the description section of the interactive query as the query and designating documents as relevant to the query if one or more instances were identified in it. The batch experiments set out to determine a baseline performance and one with maximum improvement that could be used in subsequent user experiments. Each Q-expression was used to retrieve documents from the 1991-1994 Financial Times collection (used in the Interactive Track for the past three years) for the 14 TREC-6 and TREC-7 Interactive Track topics. Average precision was calculated using the `trec_eval` program.

Results

Table 1 shows the results of our batch experiments using TREC-6 and TREC-7 Interactive Track data. The first column shows average precision, while the next column gives the percent improvement over the baseline, which in this case was the BB-ACB-BAA (basic vector space TF*IDF) approach. The baseline was improved upon by other approaches shown to be effective in other TREC tasks (e.g., ad hoc), in particular pivoted normalization (second and third rows - with slope of pivot listed in parentheses) and the Okapi weighing function (remaining rows). The best improvement was seen with the AB-BFD-BAA measure, a variant of the Okapi weighing function, with an 81% increase in average precision. This measure was designated for use in our user experiments.

	Average precision	% improvement
BB-ACB-BAA	0.2129	0%
BD-ACI-BCA (0.5)	0.2853	34%
BB-ACM-BCB (0.275)	0.2821	33%
AB-BFC-BAA	0.3612	70%
AB-BFD-BAA	0.3850	81%
AB-BFE-BAA	0.3517	65%
AB-BFF-BAA	0.3287	54%
AB-BFG-BAA	0.3833	80%
AD-AFD-BAA	0.2432	14%
AI-AFD-BCA	0.2523	19%

Table 1 - Average precision and improvement for batch runs on TREC-6 and TREC-7 interactive data.

Experiment 2 - Interactive searching to assess weighting scheme with real users

Based on the results from Experiment 1, the goal of our interactive experiment was to assess whether the AB-BFD-BAA (Okapi) weighting scheme provided benefits to real users in the TREC interactive setting. We performed our experiments with the risk that this benefit might not hold for TREC-8 interactive data, but as seen in Experiment 3 below, this was not the case.

The OHSU TREC-8 experiments were carried out according to the consensus protocol developed for TREC-7 Interactive Track and continued this year [12]. We used all of the instructions, worksheets, and questionnaires developed by consensus, augmented with some additional instruments, such as tests of cognitive abilities and a validated user interface questionnaire.

Methods

The performance measures used in the TREC-8 interactive track were instance recall and instance precision. The searcher was instructed to look for instances of each topic. Relevance assessors at NIST defined the instances from pooled searching results from all experimental groups. Instance recall was defined as the proportion of true instances identified during a topic, while instance precision was defined as the number of documents with true instances identified divided by the number of documents saved by the user.

Both the baseline and Okapi systems used the same Web-based, natural language interface shown in Figure 2. MG was run on a Sun Ultrasparc 140 with 256 megabytes of RAM running the Solaris 2.5.1 operating system. The user interface accessed MG via CGI scripts which contained JavaScript code for designating the appropriate weighting scheme and logging search strategies, documents viewed (title displayed to user), and documents seen (all of document displayed by user). Searchers accessed each system with either a Windows 95 PC or an Apple PowerMac, running Netscape Navigator 4.0.

Librarians were recruited by advertising over several librarian-oriented listservs in the Pacific Northwest. The advertisement explicitly stated that we sought information professionals with a library degree and that they would be paid a modest honorarium for their participation. Graduate students were recruited from the Master of Science in Medical Informatics Program at OHSU. They had a variety of backgrounds, from physicians or other health care professionals to having completed non-health undergraduate studies.

The experiments took place in a computer lab. Each session took three and one-half hours, broken into three parts, separated by short breaks: personal data and attributes collection, searching with one system, and searching with the other system. The personal data and attributes collection consisted of the following steps, as described in more detail in the track plenary paper [12]:

1. Orientation to experiment (10 minutes)
2. Collection of Demographic/Experience data listed in Table 2 (10 minutes)
3. Collection of Cognitive data listed in Table 2 (40 minutes)
4. Orientation to searching session and retrieval system, with demonstration of a search (10 minutes)
5. Practice search using a topic from a previous interactive track (10 minutes)

The cognitive data was obtained by using tests from the Educational Testing Service (ETS) shown in past IR research to be associated with some aspect of successful searching.

Each participant was assigned to search three queries in a block with one system followed by three queries with the other system. A pseudo-random approach was used to insure that all topic and system order effects were nullified. (A series of random orders of topics with subject by treatment blocks were generated (for balance) and used to assign topics.) Table 2 shows a sample subject-block-topic assignment.

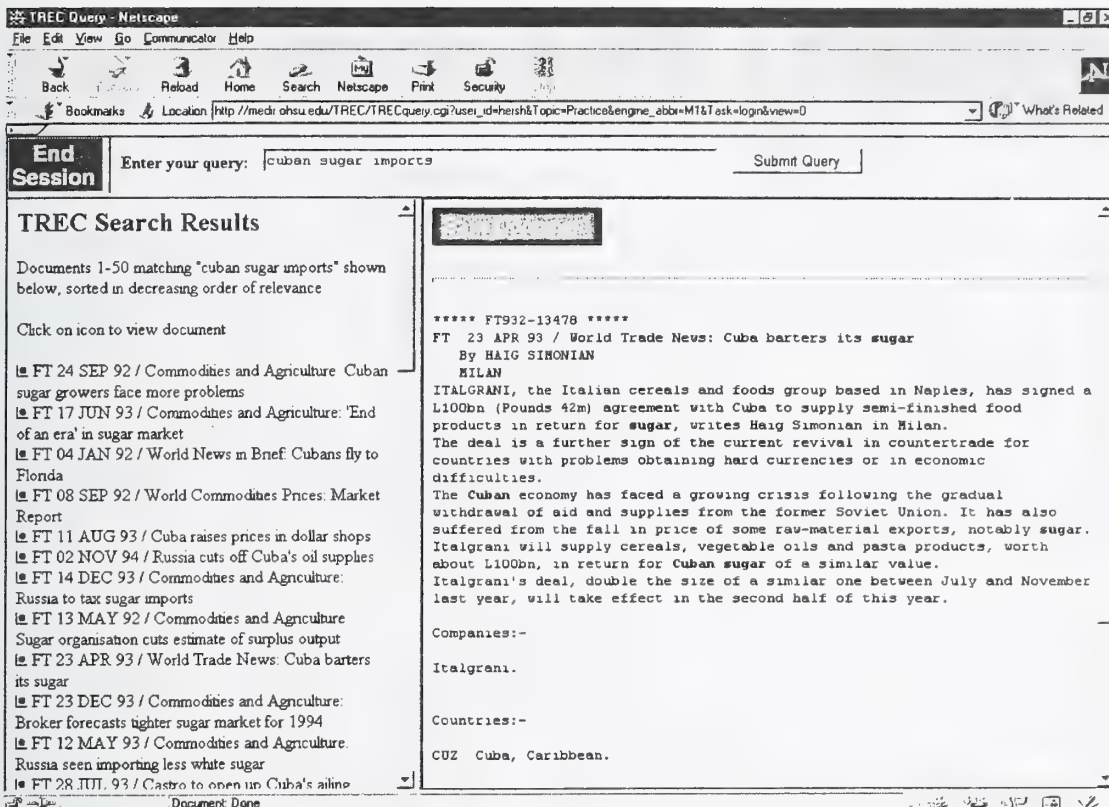


Figure 2 – Searching interface.

Subject	Block #1	Block #2
1	System 1: 6-1-2	System 2: 3-4-5
2	System 2: 1-2-3	System 1: 4-5-6
3	System 2: 2-3-4	System 1: 5-6-1
4	System 2: 3-4-5	System 1: 6-1-2
5	System 1: 4-5-6	System 2: 1-2-3
6	System 1: 5-6-1	System 2: 2-3-4
7	System 2: 6-1-2	System 1: 3-4-5
8	System 1: 1-2-3	System 2: 4-5-6
9	System 1: 2-3-4	System 2: 5-6-1
10	System 1: 3-4-5	System 2: 6-1-2
11	System 2: 4-5-6	System 1: 1-2-3
12	System 2: 5-6-1	System 1: 2-3-4

Table 2 – Sample subject-block-topic assignment for users.

The personal data and attributes collection was followed by a 10 minute break. The searching portion of the experiment consisted of the following steps:

1. Searching on first three topics with assigned system using searcher worksheet and post-topic questionnaire (60 minutes)
2. Post-System questionnaire for system used on first three topics (5 minutes)
3. Break (15 minutes)
4. Searching on second three topics with assigned system using searcher worksheet and post-topic questionnaire (60 minutes)
5. Post-System questionnaire for system used on second three topics and exit questionnaire (10 minutes)

Per the consensus protocol, each participant was allowed 20 minutes per query. Participants were instructed to identify as many instances as they could for each query. They were also instructed for each query to write each instance on the searcher worksheet and save any document associated with an instance (either by using the "save" function of the system or writing its document identifier down on the searcher worksheet).

The exit questionnaire was augmented from the consensus protocol to include the Questionnaire for User Interface Satisfaction (QUIS) 5.0 instrument [13]. QUIS provides a score from 0 (poor) to 9 (excellent) on a variety of user factors, with the overall score determined by averaging responses to each item. QUIS was given only at the end as a measure of overall user interface satisfaction since the interfaces for the two systems were identical.

An analysis of variance (ANOVA) model was fit to instance recall for these data. The factors in the model included type of searcher, the individual ID (nested in type), system, and topic. In the analysis, ID and topic were random factors, while type and system were fixed factors. Two-factor interactions (among system, topic, and type) were also included in the analysis. Residuals were examined for deviations from normality. All analyses were run in Version 6.12 of SAS for Windows 95.

Results

A total of 24 searchers consisting of 12 librarians and 12 graduate students completed the experiment. The average age of the librarians was 43.9 years, with seven women and five men. The average age of the graduate students was 36.5 years, with eight women and four men. All searchers were highly experienced in using a point-and-click interface as well as on-line and Web searching.

Table 3 shows instance recall and precision comparing systems and user types. While there was essentially no difference between searcher types, the Okapi system showed an 18.2% improvement in instance recall and an 8.1% improvement in instance precision, both of which were not statistically significant. Table 4 shows the p-values for the ANOVA model. Of importance was that while the difference between the systems alone was not statistically significant, the interaction between system and topic was. In fact, as shown by Figure 3, all of the difference between the systems occurred in just one query, 414i, which is shown above in Figure 1.

	Instance Recall	Instance Precision
System		
Baseline	0.33	0.74
Okapi	0.39	0.80
Type		
Librarian	0.36	0.76
Graduate Student	0.36	0.78

Table 3 - Instance recall and precision across systems and user types

Source	P-value
System	0.226
Topic	0.0516
Type	0.914
ID(Type)	0.0516
System * Topic	0.0269
System * Type	0.0881
Topic * Type	0.108

Table 4 - Summary of analysis of variance model for instance recall

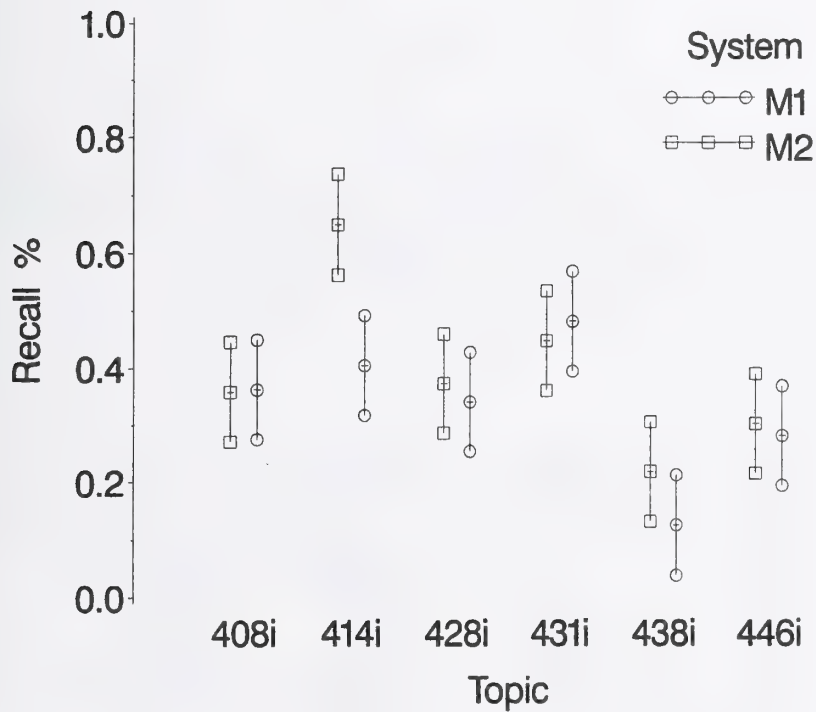


Figure 3 - Instance recall for each topic with each system (M1 = baseline, M2 = Okapi).

Experiment 3 - Verifying weighting scheme with TREC-8 data

Methods

Our final experiment consisted of verifying that the improvements in batch evaluation detected with TREC-6 and TREC-7 data held with TREC-8 data. The batch runs for the baseline and Okapi systems were repeated using the same approach of developing and using a test collection.

Results

Table 5 lists the average precision for both systems used in the user studies along with percent improvement. The Okapi AB-BFD-BAA still outperformed the baseline system, BB-ACB-BAA, but by the lesser amount of 17.6%. This happened to be very similar to the difference in instance recall noted in Experiment 2.

One possible reason for the smaller gains on the TREC-8 vs. TREC-6 and TREC-7 queries was that the average number of relevant documents for a TREC-8 query was three times higher than a query in the TREC-6 or TREC-7 sets. On average, TREC-6 interactive queries had 36 relevant documents, TREC-7 had queries 30 relevant documents, and TREC-8 queries had 92 relevant documents. The higher number of relevant documents may have given the baseline TF*IDF system a better chance of performing well, narrowing the gap between the different ranking schemes.

Also noteworthy in these results is that while query 414i achieved the second-best improvement of the six in average precision, it was far less than the improvement for 428i, which showed no improvement in the user studies. In fact, two queries showed a decrease in performance for Okapi with no difference in the user studies.

Discussion

While our experiments might be construed to suggest that retrieval systems which perform better in batch studies also do so in user studies, the actual picture is more complex. Although an improvement in the average performance was seen for a system that also performed better in batch studies, the difference was not statistically significant and occurred solely due to one query, 414i. The subject matter for this query was not markedly different from the others. The only difference was that it has far fewer relevant documents than the rest, which is likely to amplify random differences in user search strategies.

Query	Instances	Relevant Documents	Baseline	Okapi	% Improvement
408i	24	71	0.5873	0.6272	6.8%
414i	12	16	0.2053	0.2848	38.7%
428i	26	40	0.0546	0.2285	318.5%
431i	40	161	0.4689	0.5688	21.3%
438i	56	206	0.2862	0.2124	-25.8%
446i	16	58	0.0495	0.0215	-56.6%
Average	29	92	0.2753	0.3239	17.6%

Table 5 - Average precision and improvement for batch runs of TREC-8 data

Another possible interpretation of these data is that query 414i was an outlier and that differences in batch searching do not translate into better user searching. This view is supported by the large differences in the baseline and Okapi systems (positive and negative) which had no accompanying difference in the user studies.

The ultimate answer to the question of whether batch and user searching evaluations give the same results must ultimately be answered by further experiments that use a larger number of queries. The 20 queries accumulated for the Interactive Track over the last three years provides a larger base from which to start further investigations. Of course, to fully answer the question, other retrieval tasks must be represented as well, such as question-answering and high-recall situations as well.

References

1. Cleverdon CW and Keen EM, *Factors determining the performance of indexing systems*, . 1966, Cranfield UK: Aslib Cranfield Research Project.
2. Harman D. Overview of the first Text REtrieval Conference. In *Proceedings of the 16th Annual International ACM Special Interest Group in Information Retrieval*. 1993. Pittsburgh: ACM Press. 36-47.
3. Sparck-Jones K, *Information Retrieval Experiment*. 1981, London: Butterworths.
4. Meadow CT, Relevance? *Journal of the American Society for Information Science*, 1985. 36: 354-355.
5. Swanson DR, Information retrieval as a trial-and-error process. *Library Quarterly*, 1977. 47: 128-148.
6. Hersh WR, Relevance and retrieval evaluation: perspectives from medicine. *Journal of the American Society for Information Science*, 1994. 45: 201-206.
7. Lagergren E and Over P. Comparing interactive information retrieval systems across sites: the TREC-6 interactive track matrix experiment. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research And Development in Information Retrieval*. 1998. Melbourne, Australia: ACM Press. 162-172.
8. Witten IH, Moffat A, and Bell TC, *Managing Gigabytes - Compressing and Indexing Documents and Images*. 1994, New York: Van Nostrand Reinhold.
9. Zobel J and Moffat A, Exploring the similarity space. *SIGIR Forum*, 1998. 32: 18-34.
10. Robertson SE and Walker S. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM Special Interest Group in Information Retrieval*. 1994. Dublin: Springer-Verlag. 232-241.
11. Singhal A, Buckley C, and Mitra M. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM Special Interest Group in Information Retrieval*. 1996. Zurich, Switzerland: ACM Press. 21-29.
12. Hersh W and Over P. TREC-8 interactive track report. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*. 2000. Gaithersburg, MD: NIST. In press.
13. Chin JP, Diehl VA, and Norman KL. Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of CHI '88 - Human Factors in Computing Systems*. 1988. New York: ACM Press. 213-218.

Structuring and expanding queries in the probabilistic model

OGAWA Yasushi MANO Hiroko NARITA Masumi
HONMA Sakiko

Software Research Center, RICOH Co., Ltd.
1-1-17 Koishikawa, Bunkyo-ku, Tokyo 112-0002, JAPAN
{yogawa,mano,narita,honma}@src.ricoh.co.jp

1 Introduction

This is our first participation in TREC and five runs were submitted for the ad-hoc main task. Our system is based on our Japanese text retrieval system [4], to which English tokenizer/stemmer has been added to process English text. Our indexing system stores term positions, thus providing proximity-based search, in which the user can specify the distance between query terms.

What our system does is outlined as follows:

1. Query construction

The query constructor accepts each topic, extracts words in each of the appropriate fields and constructs a query to be supplied to the ranking system.

2. Initial retrieval

The constructed query is fed into the ranking system, which then assigns term weights to query terms, scores each document and turns up a set of top-ranking documents assumed to be relevant to the topic (pseudo-relevant documents).

3. Query expansion

Based on the feedback from the pseudo-relevant documents, the query expander collects and ranks the words in the pseudo-relevant documents and the words ranked the highest are added to the original query, with the words already in the query re-assigned new term weights.

4. Final retrieval

The ranking system performs final retrieval using the modified query.

In what follows, we explain what is done in each of the steps in more detail.

2 Query construction

We have employed automatic query processing to construct single-word and phrasal search terms. Our query processing involves a series of steps to identify the important concepts in each topic. In what follows, we describe in some detail how single and phrasal search terms are created by linguistic methods and represented as a structured query.

2.1 Stemming and morphological expansion of query terms

Natural language text in each topic is processed by our English tokenizer and stemmer to output stemmed word tokens. We have developed a new stemmer which conflates morphologically related words in two steps: the first step stems terms for document indexing and query processing and the second performs morphological expansion of query terms. To avoid degradation of performance by overstemming, our stemmer only stems an inflected form to its base form and a sequence of derivational suffixes to the initial suffix.¹ For example, "humanization" and "humanized" are stemmed to "humanize," but not "humanize" to "human." When constructing queries, each search term is stemmed and expanded with its morphological variants; "humanize" is possibly expanded with "human," "humanity," "humanise," and so on.

Although a run with stemmed queries without morphological expansion showed rather consistent improvements over a run without stemming, expansion of query terms produced inconsistent results. Some queries benefited a lot; others were damaged a lot. We found that both benefits and damages resulted from derivational variants. To the contrary, spelling variants rarely had ill effects although they improved only a few queries. Avoiding risks, we decided to expand terms only with spelling variants for our submitted runs. Thus, a stemmed term "humanize" is expanded to #SYN(humanize,humanise) with a synonym operator #SYN.

2.2 Single term selection

From the terms extracted by the stemmer, the query constructor selects single-word search terms for the query by eliminating very common and irrelevant words, i.e., stopwords. We have used two kinds of stopword lists, the Fox's [1] word list for the <title> field and its augmented word list we created for the <desc> field. To augment the Fox's word list, some dozens of unimportant words were manually added to the original list after examination of the <desc> fields from TREC-3 to TREC-7.

For example, the words "identify," "document," and "discuss" in "Identify documents that discuss clothing sweatshops" were added to the Fox's word list because these words provide no information about the information need.

2.3 Phrasal term selection

Syntactic phrases are recognized in the natural language text by applying the syntactic chunker LT_CHUNK developed at the Edinburgh Language Technology Group. This chunker uses the part-of-speech information provided by the tagger LT_POS and identifies boundaries of simple noun phrases which do not include prepositional or clausal post-modifiers.

Each noun phrase is tokenized/stemmed and then stripped of all stopwords. As a result, the phrases consisting of two or more single words are extracted for use in search terms.

For phrases consisting of three or more single words, we have given a special treatment because we have experimentally found out that these multi-word phrases are less likely to match documents in the collection. First, all pairs of single words are derived from the target phrase. For example, the noun phrase "industrial waste disposal" is decomposed to derive three possible word pairs such as "industrial waste," "waste disposal," and "industrial disposal." Second, the word pairs which never occur in sequence in the TREC test collections are discarded. In the above example, the last word pair "industrial disposal" is discarded by this processing.

¹Our decision to what extent we should stem a word is partly based on [3] and [2].

Further, we handle hyphenated words as phrasal terms by specifying that the constituent words be found adjacent in a document.

2.4 Query representation

Single and phrasal search terms are combined into a query using syntax of our query language. As mentioned above, term variants by morphological expansion are expressed with a synonym operator #SYN. Phrasal terms are treated as arguments to a proximity operator #WINDOW and two forms of representation are provided with different window sizes and constraints on word order.

We have also introduced a scoring operator #SCALE to phrasal term representation to adjust its term weight because our preliminary experiments suggest that a phrasal term should be given a lower term weight than a single term. After a series of experiments on weight adjustment, we have fixed two different combinations of weight scales, respectively, for phrasal terms from the <title> field only and those from the <title> and <desc> fields.

To sum, our sample queries are expressed as follows:

A query from the <title> field only:

```
#OR(industrial,waste,disposal,
    #SCALE[0.1] (#WINDOW[1,1,o] (industrial,waste)),
    #SCALE[0] (#WINDOW[2,500,u] (industrial,waste)))
```

A query from the <title> and <desc> fields:

```
#OR(killer,bee,attack,human,#SYN(africanize,africanise),
    #SCALE[0.4] (#WINDOW[1,1,o] (killer,bee)),
    #SCALE[0.25] (#WINDOW[2,500,u] (killer,bee)))
```

where #WINDOW[1,1,o] specifies that the two words be found adjacent in a document while #WINDOW[2,500,u] specifies that the two words not be found adjacent but occur within a window of 500 words with no constraint on word order. Note also that single terms are merged with phrasal terms using a logical operator #OR.

3 Initial retrieval

For each query constructed by the query constructor, the ranking system ranks the documents in the target document collection and retrieves top-ranking documents. To rank documents, the system uses term weighting and document scoring formulae similar to Okapi's but with some modifications, mostly in term weighting.

In the probabilistic model [5], each term in the query is assigned a term weight to represent the appropriateness of the term as a discriminator in the collection. Terms are weighted according to

$$w_t = \log \frac{p}{1-p} - \log \frac{q}{1-q} \quad (1)$$

where p is the probability that a document contains the term, given that it is relevant and q is the probability that a document contains the term, given that it is not relevant.

In Okapi [6], the probabilities p and q are given by

$$p = \frac{p_0}{p_0 + (1 - p_0) \frac{N-n}{N}} \quad (2)$$

$$q = \frac{n}{N} \quad (3)$$

where N is the number of documents in the collection, n is the number of the documents in which the term occurs and p_0 is the estimate of the probability that the term occurs in a relevant document when no document contains the term. From (1), (2) and (3), we have

$$w_t = \log \frac{p_0}{1-p_0} + \log \frac{N}{N-n} - \log \frac{n}{N-n}. \quad (4)$$

By replacing $\log \frac{p_0}{1-p_0}$ with k_4 , we have

$$w_t = k_4 + \log \frac{N}{N-n}$$

where $-\infty < k_4 < \infty$ since $0 \leq p_0 \leq 1$.

Now, with this weighting formula, k_4 is less than zero when p_0 is estimated as smaller than 0.5, which is usually a reasonable estimate. However, as has been pointed out, with the value of k_4 negative, the term weight could result in a negative value depending on the value of n , the consequence of which would be degenerate retrieval.

To solve this problem, we have changed the way how the probability p is estimated. That is, in our modified formula, p is estimated as

$$p = p_0 + (1-p_0) \frac{n}{N}. \quad (5)$$

From (1), (5) and (3), we have

$$w_t = \log \left(\frac{p_0}{1-p_0} \cdot \frac{N}{N-n} + \frac{n}{N-n} \right) - \log \frac{n}{N-n} \quad (6)$$

and if we let k'_4 be $\frac{p_0}{1-p_0}$, we have

$$w_t = \log \left(k'_4 \cdot \frac{N}{N-n} + 1 \right).$$

Note that with our formula, the value of k'_4 never gets negative regardless of the value of p_0 , thus ensuring that the term weights are always positive. By keeping the term weights positive, the quality of retrieval is maintained even in the worst case.

With each term weighted according to the above formula, the ranking score for each document is calculated using a formula very similar to Okapi's.

$$s_{d,q} = \sum_{t \in q} \frac{w_t}{k_4 + \log N} \cdot \frac{f_{t,d}}{k_1((1-b) + b \frac{d_t}{d_{ave}}) + f_{t,d}}$$

where $f_{t,d}$ is the within-document frequency of the term, d_t is the document length d_{ave} is the average document length, k_1 and b are parameters, just as in Okapi's.

4 Query expansion

After initial retrieval, the query expander collects single terms in the pseudo-relevant documents and ranks them according to its Term Selection Value (TSV) while reweighting query terms, using formulae adopted from Okapi's and modified in three ways as described below. The top-ranking single terms are then added to the original query with their respective term weights. The single terms and phrasal terms originally included in the query are also given re-assigned term weights, multiplied with a bonus factor.

4.1 Term weighting 1

The first of the modifications in the TSV formula involves a change in term weighting after initial retrieval, from Okapi's to the one that reflects the modification in term weighting during initial retrieval mentioned above.

In Okapi, term weights are re-assigned after initial retrieval, when feedback from retrieved documents becomes available. The new term weights are calculated as a weighted average of the term weight estimated without any relevance feedback and the term weight estimated solely from relevance feedback. That is, if we put the term weight estimated without feedback, which was assigned for initial retrieval, as

$$w_t = \log \frac{p}{1-p} - \log \frac{q}{1-q} = w_p - w_q,$$

the new term weight after relevance feedback would be

$$w_t = C_p * w_p + (1 - C_p) * w'_p - C_q * w_q - (1 - C_q) * w'_q \quad (7)$$

where w'_p and w'_q are term weight components based on relevance feedback and C_p and C_q are coefficients. Specifically, with

$$w_p = \log \frac{p_0}{1-p_0} + \log \frac{N}{N-n} = k_4 + \log \frac{N}{N-n}, \quad w_q = \log \frac{n}{N-n} \quad (8)$$

from (4), the term weight after feedback in Okapi is calculated as

$$\begin{aligned} w_t = & \frac{k_5}{k_5 + \sqrt{R}} (k_4 + \log \frac{N}{N-n}) + \frac{\sqrt{R}}{k_5 + \sqrt{R}} \log \frac{r+0.5}{R-r+0.5} \\ & - \frac{k_6}{k_6 + \sqrt{S}} \log \frac{n}{N-n} - \frac{\sqrt{S}}{k_6 + \sqrt{S}} \log \frac{s+0.5}{S-s+0.5} \end{aligned}$$

where R is the number of relevant documents, r is the number of relevant documents containing the term, S is the number of non-relevant documents, s is the number of non-relevant documents containing the term, and k_5 and k_6 are parameters.

In our system, we followed the same principle as Okapi's, adopting (7). However, as described earlier, since we changed initial w_p from (8) to

$$w_p = \log \left(\frac{p_0}{1-p_0} \cdot \frac{N}{N-n} + \frac{n}{N-n} \right) = \log \left(k'_4 \cdot \frac{N}{N-n} + \frac{n}{N-n} \right)$$

as shown in (6), term weighting after feedback was changed accordingly, resulting in the formula

$$\begin{aligned} w_t = & \frac{k_5}{k_5 + \sqrt{R}} \log \left(k'_4 \frac{N}{N-n} + \frac{n}{N-n} \right) + \frac{\sqrt{R}}{k_5 + \sqrt{R}} \log \frac{r+0.5}{R-r+0.5} \\ & - \frac{k_6}{k_6 + \sqrt{S}} \log \frac{n}{N-n} - \frac{\sqrt{S}}{k_6 + \sqrt{S}} \log \frac{s+0.5}{S-s+0.5}. \end{aligned}$$

4.2 Term weighting 2

The above term weighting formula underwent further modification when we noticed some incidents of words that are too common to be useful in a query appearing in the expansion

terms selected using the formula. To keep these words from being included in expansion, the term weighting formula was changed to reflect the document frequency of the term in the coefficient part of the formula as

$$w_t = \frac{k_5}{k_5 + \sqrt{\frac{R}{R+n-r}}} \log \left(k'_4 \frac{N}{N-n} + \frac{n}{N-n} \right) + \frac{\sqrt{\frac{R}{R+n-r}}}{k_5 + \sqrt{\frac{R}{R+n-r}}} \log \frac{r+0.5}{R-r+0.5} \\ - \frac{k_6}{k_6 + \sqrt{\frac{S}{S+n-s}}} \log \frac{n}{N-n} - \frac{\sqrt{\frac{S}{S+n-s}}}{k_6 + \sqrt{\frac{S}{S+n-s}}} \log \frac{s+0.5}{S-s+0.5},$$

thus reducing the relative weight of terms appearing in both the pseudo-relevant documents and the whole document collection. (In the experiments, S was set to 0.)

4.3 Term Selection Value

We also looked at the whole TSV formula of Okapi's [7]

$$TSV = (r/R - \alpha \cdot s/S) \cdot w_t$$

where α is a parameter. From our observation on the terms selected with this formula, however, we felt that it would be better to eliminate those terms too specific to serve as expansion terms, such as a telephone number, that were included in the selection. To that end, the Okapi's TSV formula was modified to reflect the within-document frequency as

$$TSV = \left(\sum_{d \in R} \frac{f_{t,d}}{k_1((1-b) + b \frac{d_t}{d_{ave}}) + f_{t,d}} \bigg/ R - \beta \cdot \sum_{d \in S} \frac{f_{t,d}}{k_1((1-b) + b \frac{d_t}{d_{ave}}) + f_{t,d}} \bigg/ S \right) \cdot w_t$$

where β is a parameter, which would lower the TSV of those terms that occur only once or twice in a document, in comparison with the terms that occur more often per document. (We also required r to be larger than 1 for a term to be considered. Also, S was set to 0 in the experiment as in term weighting.)

With these modifications, our expansion method is more likely to select terms neither too common throughout the collection nor too rare to be applicable to the whole collection, without resorting to such cut-off measures as a stopword list or a filter to exclude, for instance, words containing digits.

5 Final retrieval

The expanded-and-reweighted query is sent to the ranking system and documents are retrieved as final result. Document ranking is done just as in initial retrieval, except that the term weights are supplied by the query.

6 Results

We produced eight runs using different combinations of the following conditions of the queries:

- Queries using only <title> and queries using <title> and <desc>

- Queries using no phrasal search terms and queries using phrasal search terms
- Queries using no expansion and queries using expansion

We tuned up the formulae using mainly queries generated from the TREC-7 topics. For runs with expansion, title-only queries were mostly used for the tuneup. Parameters we chose for each of the eight runs are listed in Table 1 – 3. Note that in Table 1 and Table 2, we chose different sets of parameter values for the same retrieval parameters. This is because, for retrieval in a run with no query expansion, we wanted parameter values that would maximize average precision, whereas for initial retrieval for a run with query expansion, we looked for parameter values that would maximize precision at ten retrieved documents.

Table 1: Parameters for runs without expansion

	no phrases		phrases	
	title only	title+desc	title only	title+desc
k_1	0.75	1.00	0.50	0.75
b	0.25	0.25	0.25	0.25
k'_4	0.20	0.20	0.05	0.05
Scale for #WINDOW[1,1,o]	–	–	0.10	0.40
Scale for #WINDOW[2,500,u]	–	–	0.00	0.25

Table 2: Parameters for runs with expansion (initial retrieval)

	no phrases		phrases	
	title only	title+desc	title only	title+desc
k_1	0.50	1.00	0.75	1.00
b	0.25	0.25	0.25	0.25
k'_4	0.50	0.10	0.20	0.20
Scale for #WINDOW[1,1,o]	–	–	0.10	0.25
Scale for #WINDOW[2,500,u]	–	–	0.10	0.10

Table 3: Parameters for runs with expansion (final retrieval)

	no phrases		phrases	
	title only	title+desc	title only	title+desc
Number of documents used for expansion	10	10	10	10
k_1	0.75	1.00	0.75	0.75
b	0.25	0.25	0.25	0.25
k'_4	0.20	0.20	0.10	0.05
k_5	0.25	0.25	0.25	0.25
Scale for #WINDOW[1,1,o]	–	–	0.40	0.40
Scale for #WINDOW[2,500,u]	–	–	0.25	0.25
Maximum number of terms to be added	25	30	25	30
Minimum number of terms to be added	10	10	10	10
Minimum r for term to qualify	2	2	2	2
Bonus factor for query terms	3.5	4.0	3.5	4.0

The experimental runs using the above parameters resulted in the following average precision measurements (Table 4). The table clearly shows improvement in performance when phrasal search terms are used and when queries are expanded, especially for queries using both <title> and <desc>.

Table 4: Average precision for TREC-7 topics

	title only	title + desc
no phrases/no expansion	0.2033	0.2127
phrases/no expansion	0.2120	0.2373
no phrases/expansion	0.2513	0.2571
phrases/expansion	0.2584	0.2838

Using the same parameters as above, the results for the TREC-8 topics are shown in Table 5.

Table 5: Average precision for TREC-8 topics

	title only	title + desc
no phrases/no expansion	0.2560	0.2363
phrases/no expansion	0.2572	0.2633
no phrases/expansion	0.2647	0.2426
phrases/expansion	0.2689	0.2748

In TREC-8 runs, the effect of expansion was not as great as in TREC-7 runs as far as the above results are concerned. This may be because the parameter values we experimented with were not optimal. In fact, by re-adjusting the parameters, especially the bonus factor for query terms, we saw improvement in the average precision – for example, 0.2800 for title-only queries with no phrases.

References

- [1] C. Fox. A stop list for general text. *ACM SIGIR Forum*, Vol. 24, No. 2, pp. 19–35, 1991.
- [2] D.A. Hull. Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, Vol. 47, No. 1, pp. 70–84, 1996.
- [3] R. Krovetz. Viewing morphology as an inference process. In *Proc. of 16th ACM SIGIR Conf.*, pp. 191–203, 1993.
- [4] Y. Ogawa and T. Matsuda. An efficient document retrieval method using n-gram indexing (in Japanese). *Transactions of IEICE*, Vol. J82-D-I, No. 1, pp. 121–129, 1999.
- [5] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, Vol. 27, pp. 129–146, 1976.
- [6] S.E. Robertson and S. Walker. On relevance weights with little relevance information. In *Proc. of 20th ACM SIGIR Conf.*, pp. 16–24, 1997.
- [7] S. Walker et al. Okapi at trec-6: Automated ad hoc, vlc, routing, filtering and qsdr. In *Proc. of 6th Text REtrieval Conf. NIST*, 1996.

The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC 8

Michael Fuller* Marcin Kaszkiel* Sam Kimberley* Corinna Ng*[†] Ross Wilkinson[†] Mingfang Wu*[†]
Justin Zobel*

1 Ad-hoc Task

1.1 Background

The focus of our work in TREC 8 has again been on the retrieval of documents using arbitrary passages. This year the system has been refined to include variable sized passages and pivot normalisation. Passage based automatic relevance feedback has also been explored, albeit without the use of negative feedback.

1.2 Method

As in previous years, an in-house version of the MG retrieval system was used for all experiments. For document ranking, documents and queries were matched using the Okapi similarity measure [13]:

$$sim(q, d) = \sum_{t \in q \wedge d} w_{d,t} \cdot w_{q,t} \quad (1)$$

where

$$w_{d,t} = \frac{(k_1 + 1) \cdot f_{d,t}}{k_1 \cdot [(1 - b) + b \cdot \frac{W_d}{avr_W_d}] + f_{d,t}}$$

$$w_{q,t} = \frac{(k_3 + 1) \cdot f_{q,t}}{k_3 + f_{q,t}} \cdot \log \frac{N - f_t + 0.5}{f_t + 0.5}$$

The constants k_1 , k_3 and b were set to 1.2, 1000 and 0.75 respectively, as recommended by the City University group [13]. W_d is the length of the document d in bytes and avr_W_d is the average document length in the entire collection. N is the total number of documents in the collection, f_t is the number of documents in which term t occurs, and $f_{x,t}$ is the frequency of term t in either a document d , passage p or query q .

For passage ranking, queries and passages were matched using a non-normalised version of the cosine similarity function:

$$sim(q, p) = \sum_{t \in q \wedge p} w_{q,t} \cdot w_{p,t} \quad (2)$$

where the weights were as follows:

$$w_{q,t} = (\log(f_{q,t}) + 1) \cdot \log\left(\frac{N}{f_t} + 1\right)$$

$$w_{p,t} = \log(f_{p,t}) + 1$$

Multiple passage sizes were used for *mds08a1*, *mds08a2*, and *mds08a3*, necessitating the lengthwise normalisation of their passage scores before they could be compared. The passage scores were normalised using a pivot with a slope of 0.2 [11].

$$sim_n(q, p) = \frac{sim(q, p)}{(1 - slope) + slope \cdot \left(\frac{W_p}{avr_W_p}\right)} \quad (3)$$

where W_p is the passage's length and avr_W_p is the average length of all of the passages, both measured in words. The similarity score for each document is the maximum of the normalised similarity scores for its passages.

Automatic relevance feedback is based on the Rocchio formula [9]:

$$Q_{new} = \alpha \cdot Q_{orig} + \frac{\beta}{|R|} \sum_{r \in R} r - \frac{\gamma}{|R'|} \sum_{r' \in R'} r' \quad (4)$$

where Q_{orig} is a weighted term vector for the original query; R is the set of relevant documents; R' is the set of non-relevant documents; and r and r' are weighted term vectors for relevant and non-relevant documents respectively. The parameters α , β and γ determine the effect of the terms from the original query, relevant documents and non-relevant documents. These parameters were set as follows; $\alpha = 1.0$ and $\beta = 2.0$, negative feedback was not used rendering the γ parameter irrelevant.

Both the document and query terms were stopped and stemmed. Single terms were stemmed according to the Lovins algorithm [7], while the stop-list contained 368 terms and is the same as that used in our TREC 7 experiments [6].

1.3 Ad-hoc runs

For each of the submitted runs *mds08a1* – 3, passage sizes in the range {50, 100, 150, ..., 600} terms were used. For each passage size, all of the documents with a non-zero similarity to the query, as calculated from equation 2, were recorded. These passage scores were then normalised using equation 3 and the maximum normalised passage score for a document was used as its similarity score. The 1000 highest scoring documents were chosen as candidate documents for each query.

The *mds08a4* and *mds08a5* were experimental runs that used query expansion based on passages and passage-based re-ranking. Using a passage size of 100 terms, 20 passages were retrieved and assumed to be relevant, giving set R in

* Department of Computer Science, RMIT,
GPO Box 2476V, Melbourne VIC 3001, Australia
{msf,martin,sam,cln,ming,jz}@mds.rmit.edu.au

[†] CSIRO, Division of Mathematical and Information Science
723 Swanston St, Carlton VIC 3053, Australia
Ross.Wilkinson@cmis.csiro.au

mingfang.wu@cmis.csiro.au

[‡] On leave at Sharp Laboratories of Europe Ltd.

	Precision @10 docs	Precision @20 docs	Precision @100 docs	Average Precision	% Δ	Recall (max 4728)
<i>title</i>						
document	0.4120	0.3520	0.2058	0.2095	0.0	2507
passage-300	0.4180	0.3470	0.2018	0.2154	+2.8	2576
optimal passage (500)	0.4160	0.3480	0.1966	0.2187	+4.4	2577
var. passages (mds08a1)	0.4040	0.3490	0.2052	0.2236	+6.7	2594
query expansion (mds08a5)	0.3900	0.3470	0.2076	0.2324	+10.9	2804
<i>title + desc</i>						
document	0.4400	0.3750	0.2104	0.2395	0.0	2690
passage-300	0.4160	0.3440	0.1986	0.2308	-3.6	2683
optimal passage (200)	0.4000	0.3350	0.2002	0.2323	-3.1	2702
var. passages (mds08a2)	0.4160	0.3590	0.2026	0.2397	+0.1	2732
query expansion (mds08a4)	0.4180	0.3680	0.1986	0.2550	+6.5	2843
<i>title + desc + narr</i>						
document	0.4280	0.3720	0.2118	0.2366	0.0	2768
passage-300	0.4000	0.3290	0.1768	0.2191	-7.4	2478
optimal passage (150)	0.4080	0.3430	0.1844	0.2256	-4.6	2578
var. passages (mds08a3)	0.4000	0.3570	0.1906	0.2338	-1.2	2640
query expansion	0.3900	0.3490	0.1998	0.2534	+7.1	2809

Table 1: The effectiveness results for the described runs on the 50 TREC 8 topics.

equation 4. No documents were assumed to be irrelevant, i.e. set R' was empty. All of the terms that appeared in the text of these passages were stopped and stemmed using the Lovins algorithm. The number of passages that each of these terms appeared in and its total number of occurrences in R was calculated. The 50 terms that appeared in the most passages were selected to be included in the expanded query. Where there was contention between terms for selection it was resolved by reference to the total number of times that the terms appeared in the top 20 passages.

The documents that contributed the 20 passages were retrieved and the weights for each of the selected terms were calculated according to:

$$w_{d,t} = \frac{1 + \log(1 + \log(f_{d,t}))}{(1 - \text{slope}) + \text{slope} \times \frac{W_d}{\text{avr-}W_d}}$$

The original query terms were re-weighted using:

$$w_{q,t} = (\log(f_{q,t}) + 1) \cdot \log\left(\frac{N+1}{f_t}\right)$$

These weights were combined using the Rocchio formula of equation 4 then normalised. With the query terms re-weighted and the new terms added to the query, the query was re-run using a fixed passage size of 300 terms, and the best 1000 documents selected.

1.4 Results and Analysis

The TREC 8 results are shown in Table 1 with the submitted runs shown in boldface. Only the *mds08a3* and *mds08a4* runs were used for pool judgments.

The *document* run is intended as a base against which the other runs can be compared; it used the Okapi similarity measure. The optimal passage runs are fixed-size-passage runs that generated the highest average precision; they are shown for comparison with the variable passage length runs. The query-expansion run for the *title+desc+narr* queries used the same automatic relevance feedback as *mds08a4* and *mds08a5*.

	> Median
mds08a1	14
mds08a2	18
mds08a3	13
mds08a4	26
mds08a5	24

Table 2: The number of the 50 queries that were better than the median average precision for each submitted run.

As noted in the TREC 7 report, passage retrieval is comparable to document-based retrieval for short queries and slightly less effective for longer queries.

It is interesting to observe that although the document based runs generally have better precision at 10, 20 and even 100 documents, the passage based runs on occasion manage comparable or even superior average precision. An example of this is provided by the title runs *mds08a1* and *document*, of table 1. This effect is caused by the less precise run having superior recall, as is shown in the recall column of table 1. In most situations, where a run retrieves more relevant documents than the document-based run it also has a higher average precision.

In contrast to previous years the use of the topics' narrative component appears to detract from the system's retrieval effectiveness. This is particularly evidenced by the Okapi document based retrieval which shows a 1.2% degradation in effectiveness when the narrative is included, compared with a 17.6% improvement in TREC 7 [6].

Table 2 compares the submitted runs against the submissions of other TREC participants, it shows the number of queries for each run that achieved a higher average precision than the median. None of the runs had the best precision for any of the queries.

The use this year of variable sized passages has been worthwhile, providing an improvement in effectiveness over the use of fixed size passages. The exception to this is for title queries where it has a lower precision at 10 documents than both the passage-300 and optimal passage runs. However, it is again greater at 20 and 100 documents. Another

Query Type	Precision @10 docs	Precision @20 docs	Precision @100 docs	Average Precision	$\Delta\%$	Recall (max 206)
<i>title</i>						
document	0.2579	0.1763	0.0589	0.3452	0.0	173
passage-300	0.2684	0.1684	0.0547	0.3356	-2.8	177
var. passages	0.2895	0.1763	0.0579	0.3451	0.0	177
query expansion	0.2684	0.1684	0.0605	0.3549	+2.8	181
<i>title + desc</i>						
document	0.2579	0.1711	0.0600	0.3305	0.0	183
passage-300	0.2368	0.1711	0.0574	0.3167	-4.2	185
var. passages	0.2684	0.1763	0.0589	0.3429	+3.8	184
query expansion	0.2789	0.1763	0.0632	0.3832	+15.9	184
<i>title + desc + narr</i>						
document	0.2684	0.1658	0.0621	0.3273	0.0	183
passage-300	0.2579	0.1684	0.0626	0.3490	+6.6	187
var. passages	0.3105	0.1816	0.0568	0.3810	+16.4	186
query expansion	0.3263	0.2079	0.0684	0.4330	+32.3	205

Table 3: The results of running the TREC-8 runs on the FR collection alone, using the 19 queries that had relevant answers in the FR collection.

benefit of this approach is that it is independent of query length; in contrast, different sizes of fixed passages are better suited to queries of different lengths, with the optimum passage size shrinking as query length increases.

The experimental *mds08a4* and *mds08a5* runs did not use variable size passages. These runs are more effective than the fixed size passage runs on which they were based. This is particularly true for longer queries, with the exception of the title+desc+narr run which registered a precision at 10 documents which was worse than both of the fixed size passage runs.

Table 3 shows how the same runs fared on the Federal Register collection (FR) alone. The documents contained in the FR collection are statutory rules and regulations of the US Federal Government, these documents are characterised by an average length that is greater than the other collections that comprise the TREC 8 collection. Only those topics that had a relevant document in the FR collection were used; this meant that there were 19 topics with 206 relevant documents.

Passage-based retrieval seems to be particularly effective for the FR collection. Whilst the effectiveness of passage-based retrieval decreased — relative to document based retrieval — for longer queries over the entire collection, it increased when only the FR collection was used. In fact, passage-based retrieval improves on document-based retrieval for virtually all of the runs. It is particularly interesting that the variable size passage runs have better recall and precision than the document-based runs, this appears to be for a number of reasons.

Firstly, running the 19 FR queries over the total collection, as opposed to the 50 queries that generated the results in table 1, gives results that are similar to the results of table 3 in that passage retrieval outperforms document retrieval. This suggests that either some of the 19 FR queries are particularly susceptible to passage retrieval or that some of the other 31 queries are poor candidates for passage retrieval.

Secondly, given the longer documents in the FR collection it would be expected for the effectiveness of passage-based retrieval to improve relative to document retrieval.

Thirdly, it is conceivable that the limited number of relevance judgments, relevant documents and candidate documents are skewing the results.

2 Question & Answer track

2.1 Introduction

We participated in the 250 byte category of the question and answer track, submitting one run, *mds08q1*. Our objective in participating in this track was to determine the appropriateness of applying traditional document retrieval techniques to the retrieval and extraction of small, focused text segments.

2.2 Method

It was our goal to determine whether the passages of text that exhibited the closest correspondence to the question in terms of content also contained a correct answer.

Therefore, with this in mind our approach was as follows;

1. Modify the question into an acceptable query for the system.
2. Retrieve the 50 most relevant passages from the document collection using passage ranking.
3. Find the 5 best sentences from the 50 retrieved passages.
4. If a sentence was over 250 bytes in length, use a 250 byte sliding window to find the segment that best matched the query.

Query generation

Queries were generated from the official questions by stopping terms and removing all punctuation. This converted the question to a conventional query that was used as input to the MG retrieval system. Generally, the resulting query was very short, with an average length of 4.8 words in the training queries and 5.3 words in the actual queries.

Extracting the best passage

These queries were used as input to the MG passage retrieval system which used the similarity measure defined in equation 2 to retrieve the most relevant passage from the 50 most relevant documents for each query. The passages were 150 words in length with each new passage commencing at a 25 word

interval from its predecessor. Because all of the passages are the same length there was no need for passage length normalisation. The document and query terms were case-folded, and stemmed using the Lovins algorithm.

Finding the best sentences

The passages retrieved by MG were segmented into individual sentences and a similarity relative to the query was calculated for each sentence. The five sentences with the highest similarity scores were then selected as the candidate answers to the question. The query and sentence terms were case-folded and stemmed by the sentence retrieval engine using the Lovins algorithm.

The similarity of each sentence was calculated by:

$$\text{sim}(q, s) = \frac{1}{w_s} \sum_{t \in q} w_{s,t} \cdot w_{q,t} \quad (5)$$

where:

$$\begin{aligned} w_{s,t} &= \log(f_{s,t} + 1) \\ w_{q,t} &= \log(f_{q,t} + 1) \cdot \log\left(\frac{N}{f_t} + 1\right) \\ w_s &= \sqrt{\sum_{t=1}^n w_{s,t}^2} \end{aligned}$$

In the above equations $f_{x,t}$ is the number of times that the term appears in x , where x is either a sentence or a query, N is the total number of documents in the collection, and f_t is the number of documents in which the term appears.

Reducing a sentence to 250 bytes

Some of the sentences that were returned were in excess of the 250 byte ceiling imposed on acceptable answers. The pruning stage described here only applied to those sentences that exceeded the 250 byte limit.

A sliding window was used to identify 250 byte segments for excessively long candidate sentences. The left edge of the window was placed at the start of the sentence, with the window extending 250 bytes into the sentence. The window was then slid across the sentence in single word increments, so that the left edge of the window was always located at the start of a word. At each position a similarity score for the window was calculated using equation 5.

When the right edge of the window reached the end of the sentence the process was stopped, and the window position that generated the highest similarity score was chosen to represent the sentence. The sliding operation did not observe word boundaries at the window's right edge.

As in the previous stages all of the query and sentence terms were case-folded and stemmed using the Lovins algorithm.

2.3 Results and analysis

This approach retrieved many short sentences with a high proportion of relevant terms but not an appropriate answer. A good example of this phenomenon were headlines, which were frequently retrieved. The role of a headline is to succinctly describe the topic of its accompanying article. It is brief and can contain a large proportion of the terms used to identify a subject. Therefore it is not surprising that a question requesting specific information about an event would retrieve

the headline of an article describing the event. Unfortunately, an acceptable answer generally required more detailed information than a headline could provide.

Another concern was that sentences containing repeated occurrences of a single query term were in some cases being retrieved before sentences that referenced multiple distinct query terms. This was a concern because the sentences containing a broader coverage of the query terms were generally the more relevant sentences.

From these observations the similarity measure underwent two modifications. Firstly, the sentence normalisation procedure was modified to reduce the disproportionate number of short sentences that were being returned. Secondly, the similarity measure was changed to reward sentences that provided a good degree of query coverage, through the use of supplementary coordinate matching [16].

Floor on sentence length

The first modification to the similarity measure was to set a floor on the length of sentences for the purpose of calculating similarity. If the calculated weight for a sentence (equal to the sentence length in the case of sentences without repeated terms) was less than a floor x its weight was fixed at \sqrt{x} . This required that a short sentence be highly relevant before it was retrieved and also allowed the similarity of sentences to be adjusted according to their length. As shown in table 4, it was an effective mechanism for improving the performance of this task. Experimentation with the use of pivot normalisation found that the thresholding mechanism provided superior sentence length normalisation for this task.

This normalisation technique converted w_s , defined previously in equation 5, to:

$$w_s = \begin{cases} w_{s'}, & \text{if } w_{s'} \geq x^{1/2} \\ x^{1/2}, & \text{otherwise} \end{cases} \quad (6)$$

where:

$$w_{s'} = \sqrt{\sum_{t=1}^n w_{s,t}^2}$$

Experimentation with varying floor lengths suggested that 30 words was a reasonable length for the current document collection; see table 4.

Coordinate Matching

The information needs represented by the TREC Q&A questions require the extraction of specific information from the document collection in the form of short, precise answers. Our approach has been to attempt to identify appropriate sentences or sentence fragments. It is our contention that when using such an approach it is preferable to combine the terms in a more conjunctive manner than would be appropriate for a conventional ranked query. It is also our belief that concepts are generally only stated once within a given question, whereas a query such as a TREC topic may contain many re-statements of a single concept through the use of synonyms, and so on. These hypotheses suggest that a candidate sentence's likelihood of relevancy increases with the number of distinct query terms that it contains. Therefore, a good candidate answer should provide coverage of most if not all of the query terms and the degree of this coverage should be taken into account when calculating the similarity for a given sentence.

Answer appears in	Sentence floor length, in terms					
	0	10	20	30	40	50
1st sentence	1	6	10	8	9	9
2nd sentence	2	3	2	8	3	2
3rd sentence	2	1	0	1	3	2
4th sentence	4	2	3	1	3	1
5th sentence	1	0	3	1	1	2
not found	28	26	20	19	19	22
Score	0.102	0.219	0.325	0.336	0.328	0.298

Table 4: System performance using a range of sentence-length minimums on the 38 training questions. These judgments were not made by NIST assessors and should not be compared to the official results.

Given this consideration, coordinate matching was used to award a sentence an extra point for each distinct query term that it contained. These coverage points are in addition to the sentence's conventional cosine similarity score. This converted the similarity score defined previously in equations 5 and 6, to the form:

$$sim(q, s) = \frac{1}{w_s} \sum_{t \in q} w_{s,t} \cdot w_{q,t} + count(t \in q \wedge s) \quad (7)$$

The effect that this mechanism has on the retrieval effectiveness of the system is shown in table 7. The submitted run, *mds08q1*, used this similarity measure with a sentence floor of 30 terms. The results of the judged run are shown in table 5, and are compared against the other judged runs in table 6.

Answer appears in	mds08q1
1st sentence	71
2nd sentence	22
3rd sentence	11
4th sentence	12
5th sentence	5
Not found	77
Score	0.453

Table 5: The judged run, *mds08q1*. It used the similarity measure of equation 7 with a sentence floor of 30 terms.

	mds08q1
= Best	77
> Median	73
= Median	108
< Median	17
= Worst	77
Average Rank	5.2

Table 6: A comparison of *mds08q1* with the other question and answer runs that were judged at NIST. The average rank metric is relative to the other runs, with the rank for a topic being the position at which the run's score was placed relative to all other runs.

Subsequent to the submission of our official run, we have explored this idea further. One modification undertaken was to normalise the cosine similarity scores to a range between 0 and 1 by dividing the scores by the query weights. This meant that when the query coverage points were added, the sentences with the most distinct query terms were automatically the highest ranked sentences. The cosine score was used

to differentiate between sentences that contained the same number of query terms.

This similarity score is described by:

$$sim(q, s) = \frac{1}{w_s w_q} \sum_{t \in q} w_{s,t} \cdot w_{q,t} + count(t \in q \wedge s) \quad (8)$$

where

$$w_q = \sum_{t=1}^n (w_{q,t}^2)^{1/2}$$

As table 7 shows, supplementary coordinate matching is a very effective mechanism for increasing the precision of this task. However it is not without disadvantages.

One undesirable effect occurs when a single logical component of the query is expressed using a phrase consisting of multiple words. This effectively gives that concept a much higher weighting in the query than those that can be more briefly described.

For example, the question "The Faroes are a part of what northern European country?" after stopping becomes, "Faroes part northern European country" This embodies three separate concepts; "Faroes", "part", and "northern European country" each of which are nominally of equal importance.

However, using word based coordinate matching, the concept "northern European country" can attract up to 3 coverage points (one per word) whereas the "Faroes" concept will only attract a maximum of 1 coverage point if it appears. This biases the results towards those sentences that contain multi-word concepts. Essentially, the problem is that query coverage points are awarded for single-word terms when they should be awarded for concepts.

This phenomenon was observed on the training data, with sentences containing the correct answer to the Faroe islands question regularly retrieved before the query coverage points were awarded. However, they were replaced by sentences about northern European countries once the query coverage points were added.

This problem could be minimised through a linguistic prepping of the query to divide it into logical concepts or common phrases, and to subsequently award query coverage points equitably across identified concepts.

2.4 Conclusion

This study has been an experiment into the effectiveness of the use of statistical IR for solving the question answering problem. It has shown that statistical IR without natural language processing can be used with some effectiveness to locate and retrieve small fragments of text as answers to questions.

Answer appears in:	conventional cosine measure	including coordinate matching	incl. cosine normalisation & coordinate matching
1st sentence	57	74	75
2nd sentence	23	18	22
3rd sentence	13	13	11
4th sentence	11	11	12
5th sentence	7	5	2
Not found	87	77	76
Score	0.389	0.460	0.470
Difference	0.0%	+18.3%	+20.8%

Table 7: Comparison of the performance of three similarity measures on the 200 Q&A questions. All of the runs shown used a sentence floor of 30 words. These runs have not been officially judged and should not be compared to the official run scores from NIST.

Simple question answering can be reasonably supported by traditional IR techniques however for more robust solutions to complex questions it is envisaged that natural language processing techniques will be required. Techniques such as coreference resolution, entity detection and question analysis have been demonstrated by groups to be effective in this task.

3 Web track

3.1 Introduction

The MDS group participated in the small web track, submitting three runs; a content-only run, *mds08w1*, and two content-and-link runs, *mds08w2* and *mds08w3*.

Our objective in participating in this track was twofold. Firstly, to determine whether simple manipulation of linking information would enable effective re-ranking of documents within a result set. Secondly, to examine the effectiveness of content-only retrieval on web data.

3.2 Content only run

This run, *mds08w1*, was performed using a similar procedure to the runs in the ad hoc task. The in-house version of MG was used to retrieve the most relevant documents using passage similarities that were calculated from the similarity measure defined in equation 2. Passages of 150 words were used, with each passage starting at 25 word intervals. It was not necessary to normalise the passage similarities as all of the passages were the same length.

3.3 Content and link runs

Our retrieval system made use of the sibling relationship between documents, where sibling documents are defined as two or more documents that are linked from the same document. This is based on the hypothesis, that if document P links to document A, then other documents that are directly accessible from document P are likely to contain similar content to document A. Therefore, it should be possible to infer some degree of related content from a sibling relationship between two documents. Of course, this will not always be the case: for example, an individual's home page may have pointers to many different fields of interest that are wholly unrelated. Therefore, an additional constraint was imposed on the sibling documents before they were used to infer related content. This constraint was to only recognise a sibling relationship if both of the siblings were retrieved in the top n documents

for a given query. This ensures that the two documents have reasonably similar content which, when combined with their sibling relationship, suggests that they share very similar content. Therefore, if A is relevant and B, A's sibling, has similar content to A then it is likely that B is relevant as well.

Sibling relationships were only identified if the siblings and the parent that links to them were all present in the WT2G collection.

There were two types of content-and-link runs used; a very simple sibling relationship implementation, and another version that aimed to overcome some of the simpler run's shortcomings.

Simple run

The run described in this section was submitted as *mds08w2*, and was processed as follows.

1. Retrieve the best 1000 documents using the same mechanism as the content-only run, call this set R .
2. For each retrieved document, d :
 - (a) Locate the document's siblings.
 - (b) For each located sibling:
 - i. If the sibling is in R add $1/k$ th of its content similarity score to the document d 's similarity score.
3. Re-rank the documents.

Therefore, the siblings of a document with a high similarity score receive a greater similarity increase than the siblings of documents with low similarity scores. The k parameter was set to 50, a figure that gave reasonable results in training.

The similarity measure can be expressed as:

$$sim(q, d) = sim_c(q, d) + \frac{1}{k} \sum_{d' \in sib(d) \wedge ret(q)} sim_c(q, d') \quad (9)$$

where, $sim_c(q, d)$ is d 's content similarity score for the query q , $sib(d)$ is the set of d 's siblings, and $ret(q)$ is the set of documents retrieved for the query q .

A concern identified from the results of this run was that very well linked documents could have their similarity score boosted enormously. In some cases, the bonus scores derived from linking completely overshadowed the base similarity score of documents with good content. This resulted in

documents with poor content that were linked to documents with good content being ranked higher than documents that themselves had good content. This is undesirable because the content similarity score is a considerably more accurate measure of relevance than the linking information score.

In *mds08w3* measures were undertaken to minimise the influence of this and other factors.

Improved Run

The run described in this section was submitted as run *mds08w3* and it was constructed as follows.

1. Retrieve the best 2000 documents using the same mechanism as the content-only run, call this set R .
2. Choose the top x documents from R , call this set X .
3. For each retrieved document, d :
 - (a) Locate the document's siblings.
 - (b) For each located sibling:
 - i. if the sibling is in X add $1/k$ th of its content similarity score to the document d 's similarity score.
 - (c) Limit the linking component of document d 's score so that it cannot exceed the contribution made by the document's content.
4. Re-rank the documents.
5. Select the top 1000 documents.

Step 1 of the algorithm extracted a larger collection of documents, 2000 as opposed to 1000 in *mds08w2*, to increase both the amount of linking information available and the pool of relevant documents. It was hoped that this would allow the identification of more sibling relationships and also improve the system's depth of recall.

Step 3(c) of the algorithm is used to reduce the accumulated impact of excessive linking on similarity scores and thereby limit linking bonuses to a secondary role. This constrained the amount that linking information could contribute to a document's similarity score to no more than the document's original score derived from its content. Therefore, a document's similarity score could be at most doubled through the existence of relevant siblings.

The other modifications to the *mds08w2* process are steps 2, and 3(b)i. These modifications are aimed at reducing the effect whereby well connected, less relevant documents tend to drive each other up the rankings. If the 950th document has the 1010th, 996th and 988th ranked documents as its siblings then its ranking will probably be significantly improved as a result of these relationships. However, at these low ranks there is only a slight probability that these documents are actually relevant. The problem is that a linking relationship with 3 relatively poor documents should not be equivalent to a relationship with a single good document, if in fact it should be worth anything at all. The motivation behind the use of linking information was to boost documents that were linked to good documents rather than documents that were linked to poor documents. It was originally intended that this problem would be handled by proportionally adjusting the sibling's score relative to the retrieved document's similarity, however this mechanism was not sufficiently restrictive to eliminate the problem described previously. In response to this, a smaller

subset of the higher ranked documents were used for sibling linking, rather than all of the retrieved documents. These higher ranked documents are much more likely to be relevant and presumably so are their siblings.

For the purposes of *mds08w3* the subset of retrieved documents that were considered for sibling linking were the 500 most highly ranked documents retrieved by the MG system.

From these modifications the similarity measure became:

$$sim(q, d) = sim_c(q, d) + sim_l(q, d) \quad (10)$$

where,

$$sim_l(q, d) = \begin{cases} sim_{l'}(q, d), & \text{if } sim_{l'}(q, d) < sim_c(q, d) \\ sim_c(q, d), & \text{otherwise} \end{cases}$$

$$sim_{l'}(q, d) = \frac{1}{k} \sum_{d' \in sib(d) \cap ret(x, q)} sim_c(q, d')$$

and $ret(x, q)$ is the set of the top x documents retrieved by the retrieval engine for the query, q .

3.4 Results and Analysis

Unfortunately, both of the submitted content-and-link runs yielded disappointing results compared to the content-only run; see table 8. On average the performance was degraded when the sibling linking information was used. The *mds08w3* run outperformed *mds08w2* as anticipated, however both were inferior to *mds08w1*. Both of *mds08w1* and *mds08w3* were judged, whereas *mds08w2* was submitted but not judged. There were queries for which the content-and-link runs outperformed the content-only run, indicating that those queries fitted the sibling model well.

Run Identification	> Median	Best
<i>Content-only</i>		
<i>mds08w1</i>	39	3
<i>Content-and-link</i>		
<i>mds08w2</i>	34	2
<i>mds08w3</i>	40	12

Table 9: Comparison of the submitted MDS runs against all of the submitted runs for the 50 queries.

The content-only result was fairly pleasing given that only a single passage size was used and no attempt was made to extract meta-information from the HTML. Interestingly, from table 9 it can be seen that although the *mds08w3* was less effective than the content-only run, its performance relative to the other submitted runs is an improvement.

Perhaps the run of greatest interest is *max-sibling* which was not submitted to TREC as it was processed post-submission. Instead of summing the similarities of a document's siblings, a proportion of the highest scoring sibling's similarity was added to the document's score; for the run shown in Table 8, this proportion was $1/10$ th. Also, rather than using the siblings of the top 500 documents, only the top 20 were used. This led to an improvement of 2.0% in average precision over the base content-only run. Whilst far from significant, it was gratifying to improve upon the content-only run after many unsuccessful attempts. The improvement gained from the inclusion of linking information in the *max-siblings* run was similar to the greatest increases observed by any of the participating groups. An improvement was also observed on the

Run Identification	Precision @5 docs	Precision @10 docs	Precision @20 docs	Precision @100 docs	Average Precision	% Δ
<i>Content-only</i> mds08w1	0.4480	0.4480	0.3860	0.1994	0.3220	0.0
<i>Content-and-link</i> mds08w2	0.4000	0.3860	0.3330	0.1894	0.2878	-10.6%
mds08w3	0.4440	0.4120	0.3590	0.2010	0.3047	-5.4%
max-sibling	0.4680	0.4360	0.3830	0.2030	0.3284	+2.0%

Table 8: The results for the small webtrack runs using the 50 TREC 8 topics, each webtrack topic used the title and description components of the query.

training data, suggesting that this approach is a preferable mechanism for the combination of this type of evidence.

4 Interactive Retrieval Track

4.1 Introduction

In TREC7, we tested using clustering technology to organize retrieved documents for aspectual retrieval, but did not find a significant gain for the clustering interface over a ranked list interface. This year, we investigated a question-driven categorization. Unlike the clustering approach, which was data-driven and attempted to discover and present topic relationships that existed in a set of retrieved documents without taking users into account, the question-driven approach tries to organize retrieved documents in a way that is close to the users' mental representation of the expected answer. In our approach, the retrieved documents are categorized dynamically into a set of categories derived from the user's question. The user determines which of several possible sets of categories should be used to organize retrieved documents.

Our participation in TREC-8 was to investigate and compare the effectiveness and usability of this question-driven classification with a ranked list model. In the following sections we present a rationale for the question-driven approach, and then describe an experiment that compares this approach with a more traditional ranked list presentation. We then report and analyze the results of this experiment. Based on these findings and discussions, we conclude with some recommendations for future improvement.

4.2 A Question-driven Approach

The nature of information needs is variable. A user may need to find specific facts, to learn about a topic, to gather a variety of information, or may simply wish to explore an information set without having a well defined-goal [1, 15]. It is difficult for an information access tool to satisfy all types of information needs with equal effectiveness. Our question driven approach mainly focuses on information needs that involve seeking specific facts about a topic. In particular, we are considering aspect queries: topics whose answer consists of more than one related piece of information. For example, the question "what non-surgical alternatives exist for treating heart disease?" might have diet, exercise, meditation, and drug programs as different aspects of the answer.

When a user seeks specific facts, they are usually able to describe the type of information they are after. Consider a user who wants to know "what countries had ferry sinking that caused 100 or more people to lose their lives?"; the answer sought consists of the names of those countries that meet the stated criteria. Thus, this user will be actively focused on

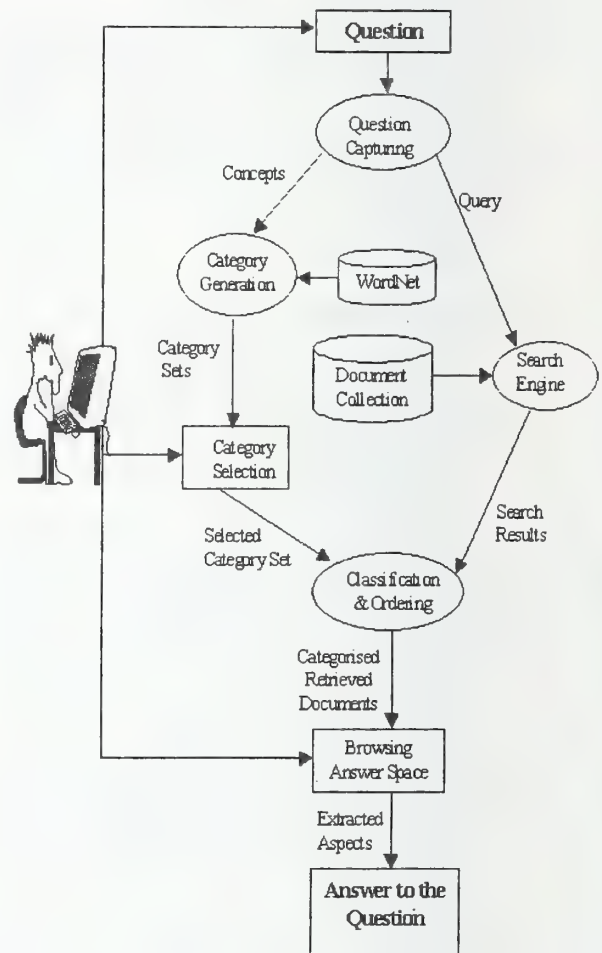


Figure 1: A Question-driven approach

searching for names of countries from retrieved documents. If the retrieved documents can be categorized by country name, this task is simplified, allowing the set of retrieved documents to be more easily analyzed.

The semantic relationship between the categories (in our example, the names of countries) and the query term from which they are derived (in our example, "country") is that of hyponym to hypernym [5]. For instance, Australia is a hyponym of country, and country is a hypernym of Australia.

An architecture for a system that categorizes retrieved document using question-driven classification is shown in Figure 1. This system contains three significant new stages: a category generation stage, a category selection stage, and a document classification and ordering stage.

Category generation. The category generator extracts keywords from each query, and uses WordNet[5] to identify a set of hyponyms for each keyword. These hyponym sets form the basis for candidate category sets. We do not attempt to distinguish between alternate senses when identifying sets of hyponym.

Category selection. For a given query, there may be multiple ways of classifying the set of retrieved documents. Automatically determining the appropriate classification axis is in itself a difficult procedure. Rather than attempting to divine users' intention, we let the user select the categorization appropriate for organizing the retrieved documents. In our implemented system, a window (shown in Figure 2) shows users the extracted keywords and their associated categories; users can consider the alternative classifications before selecting the most appropriate.

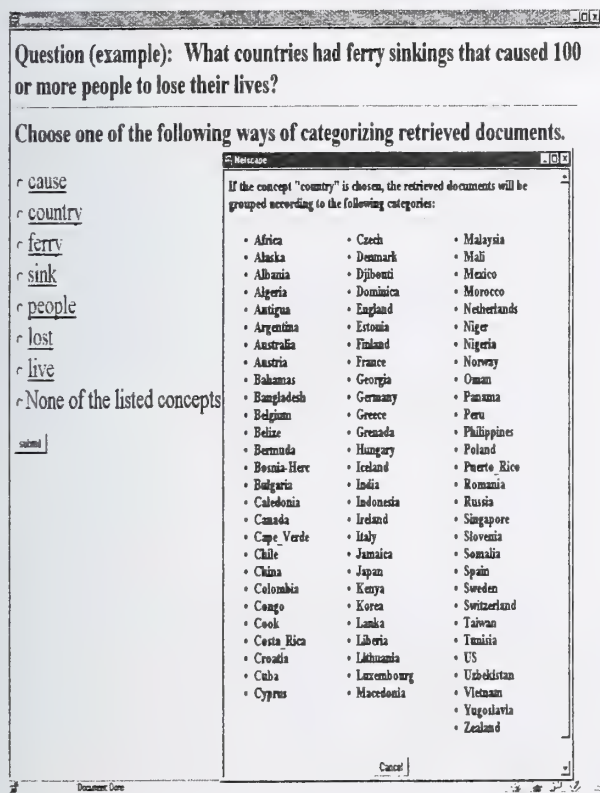


Figure 2: Interface for selecting an appropriate categorization

are then matched against the selected set of categories. Classification is based on ranking the set of retrieved documents by their similarity to the terms describing each category; in our initial implementation, each category is restricted to the ten highest-ranked documents for that category. Documents may belong to more than one category; those that do not match any specific category are allocated to a category of miscellaneous documents. Within each category, the documents are ordered according to their similarity to the original query. Overall, categories are ranked by the similarity of their first-ranked document to the query.

After the search results have been categorized and ranked, they are presented to a user as shown in Figure 3. The interface is divided into halves. In the left half, the upper frame shows the document categories. Each category is expandable and collapsible; in Figure 3 the first category is shown collapsed, and the second expanded. The middle frame shows the already discovered aspects, along with the saved documents relevant to each aspect. A button in the bottom frame enables users to add new categories into which documents may be classified. When any document is selected from the upper-left or middle-left frame, its content is shown in the right half of the window. Any terms that match the currently expanded category are highlighted in red; terms that match the descriptions of other categories are highlighted in blue. This highlighting is intended to help users more easily locate potential answers from within what may be lengthy documents. When the user finds information relevant to an aspect of the topic in a document, they can click on "Save Instances" button. This causes a pop-up window to appear in which the user can note the aspects to which the document is relevant. The discovered aspects and their associated document are then added to the middle-left frame. Whereas the upper-left frame helps the user to search for information that can contribute to their answer, the information in the middle-left frame helps the user synthesize their answer.

4.3 Experimental Setup

Goal. An experiment was conducted to evaluate the usefulness of question-driven classification. The experiment was intended to investigate, for a fact finding task using aspect queries, the ability of user directed, dynamic categorization of retrieved documents to:

- help users find more aspects relevant to their question;
- enhance user satisfaction.

A system using a ranked list interface was used as a control.

Interfaces. The question-driven classification interface is shown in Figure 3. The ranked list interface is shown in Figure 4. As the experiment was focused on comparing alternative organizations of retrieved documents, the two interfaces were kept as consistent as possible, where appropriate. The interfaces varied in three ways. One, the classification-based interface contained a list of expandable categories of retrieved documents in the upper-left frame, whereas the ranked list interface contained a simple ranked list of retrieved documents. Two, the classification-based interface allowed users to interactively add additional categories. Three, no term highlighting was used when displaying documents using the ranked list interface.

Retrieval Engine. The MG[16] search engine with an implementation of passage based retrieval and a variant of the

Classification & ordering. The retrieved documents

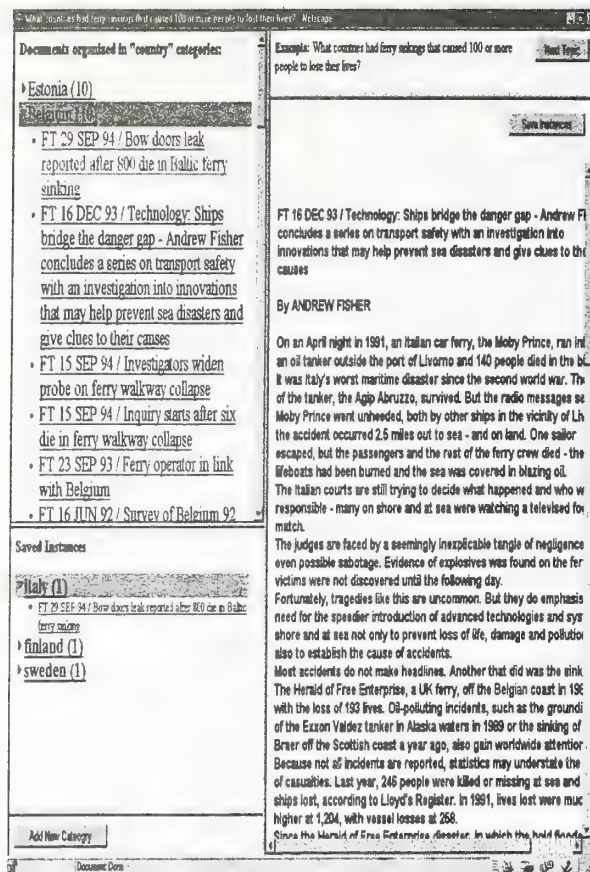


Figure 3: The interface for categorization

Okapi similarity measure[6] were used as a retrieval mechanism. Those 300 top ranked documents for each topic were then available for display as a list or via categorization. Our previous experiments on TREC-7 aspect topics showed that the top 300 documents on average contained almost 90% of available topic aspects.

Experimental design and procedure. Twenty four subjects undertook the experiment, according to the Latin Square arrangement stipulated by the TREC-8 Interactive Track guidelines. All subjects were either undergraduate or master student from the department of computer science, RMIT, recruited via an internal RMIT newsgroup. All subjects are male, had an average age of 23, 3 years on line search experience, and average FA-1 (Controlled Associations) score of 28.6 and VZ-1 (paper folding) score 15. None of the subjects had previously participated in any TREC experiment.

When subjects arrived at the experiment site, they first filled in a pre-search questionnaire and completing two psychometric tests: FA-1 and VZ-1. Subjects were then given a quick demonstration of the main functions of each interface. During the experiment, prior to using a system for the first time, subjects attempted an example topic to familiarize themselves with its interface; they were free to ask questions about the interface at this point. Each subject was required to fill in post-topic questionnaires after completing each topic, post-system questionnaires after completing their three allocated topics on each system, and an exit questionnaire at the conclusion of the experiment. Subjects were permitted

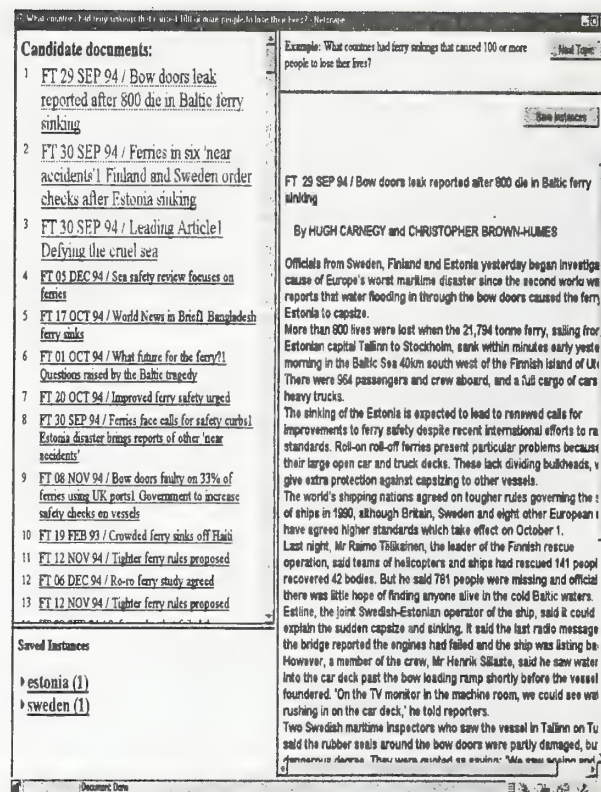


Figure 4: The interface for ranked list

up to twenty minutes to complete each topic; at the twenty minute mark they were informed that the time allocated had expired and were directed to complete their current action, complete the appropriate questionnaire, and move on to the next topic. However, all actions time-stamped outside the allocated twenty minutes were discarded for evaluation purposes. During each search session, every "significant" event such as a user selecting a classification scheme, a category, a document, an interface button, or entering text - was automatically logged and time-stamped. Participants were aware only the differences between the two interfaces; they were not informed which interface was the control system and which was the experimental system.

4.4 Results and Discussion

Effectiveness

In the interactive track, system performance is mainly measured in terms of aspectual precision and aspectual recall, where the aspectual judgements is made by independent assessors. The assessors' judgement was taken as an objective assessment of the quality of the documents that were chosen for viewing. Given that users are involved, aspectual judgement can also made by users (when they select/save an aspect). The subjects' judgement reflects their subjective conception of document's relevance to the topic in terms of their own understanding of the information need. We intend to compare the performance of two interfaces using both subjects' and assessors' judgements.

Subjects' judgement. We started our initial evaluation based on the number of aspects saved by subjects. Here, the relevance of the aspects was solely determined by each individual subject. Table 10 shows the average number (mean) of saved aspects for each topic.

	408	414	428	431	438	446	Average
L	8.5	6.6	8.3	8.9	13	7.6	8.8
C	10.1	7.3	8.8	11.3	11.6	6.7	9.3

Table 10: The average number of saved aspects for each topic in Subjects' view (L: ranked list interface; C: classification-based interface)

Table 10 shows that, overall, subjects saved more aspects by using the classification-based interface than the list interface. The mean number of saved aspects is 9.3 ($SD = 5.2$) for the classification-based interface, and 8.8 ($SD = 4.1$) for the ranked list interface; this difference is not statistically significant.

Before the experiment, we anticipated that for the four "country" topics (414, 428, 438, and 446), where instances or aspects can be differentiated by country, a classification-based interface would work better than a ranked list, as the retrieved documents are exactly organized under possible aspects; for the other two "non-country" topics (408 and 431), the performance of the classification-based interface would be uncertain, as the categorizations available to the users did not match well with a reasonable division of the topics into aspects. Table 10 shows that, for four topics, more aspects were saved using the classification-based interface than the ranked list interface. That the opposite occurred for topics 438 and 446 was unexpected.

Examination of the data and session log files revealed that the categories for topic 438 were not ranked correctly due to a bug in the code (fortunately the other 5 topics were not affected), as a result, the most relevant categories were not ranked on the top. Therefore, we have excluded this topic from any further evaluation.

We also noticed that five of the twelve subjects did not choose "country" as the basis for classification for the topic 446. It may be that some subjects had difficulty understanding the information need represented by the topic.

There was no correlation between the results of FA-1 and VZ-1 tests and the number of aspects saved by subjects.

Assessors' judgement. The distribution of the assessed aspects for each topic in the pool of candidate documents is shown in Table 11 and Table 12.

Two sets of lists were extracted from the experiment logs: the ordered lists of documents whose full text was viewed during each search, or the "read" lists; and the ordered lists of documents from which subjects saved at least one aspect, or the "saved" lists. Table 13 presents the average number of documents read/saved from each system, and aspectual precision and recall for each list under each system.

Table 13 shows that, on the average, subjects read more documents using the classification-based interface, but the documents comprising the classification-based interface read list on average contained fewer aspects than those from the list interface. Subjects saved approximately the same number of documents from the classification-based interface, but again, these sets of saved documents do not cover as many aspects as those from the list interface, although the difference on aspectual recall between these two interfaces is not

significant.

We had expected that the classification-based information organization of the retrieved documents would do better because we believe this organisation is closer to users' expectation of the answer. We think the following issues could explain the reason why the classification-based interface didn't perform better than the list interface:

- For the country topics 414, 428 and 446, the average aspectual recall for the top 20 documents of the ranked list is 0.516. To outperform the basic ranked list, the categorization approach would need to offer the user a reasonable viewing sequence with a higher recall. However, the category ranking, as implemented, failed to provide this. For example, consider the following strategy: if a user were to view the highest ranked, previously unread document from each of the first 20 ordered categories in turn, the average aspectual recall for this list of 20 documents is only 0.509. If the user modified this strategy by skipping categories for which they had already found relevant aspects (in a previously read document), the average aspectual recall drops further to 0.458.
- Many documents appear in more than one category. For topics 414, 428 and 446, each subject expanded 14.3 categories on average, whereas each subject read 18.7 documents on average, indicating that typically more than one document was selected from each category. But for many of the first 20 categories from each topic (414: 6/20; 428: 9/20; 446: 0/20), the second ranked documents in each category contributed no additional aspects. This may indicate that while the classification-based interface is suitable for organizing answers and for finding specific aspects, it is not very effective for the TREC-8 task of finding non-repeated aspects.

Evaluation of the experiment based on the subjects' judgement presents a clearly different picture to evaluation based on the assessors' judgement. This suggests that the subjects' understanding of the sought-after aspects is slightly different from the assessors' view, or that the subjects did not fully understand what was required for some topics. For example, for Topic 428 ("What countries other than the US and China have or have had a declining birth rate"), there is confusion over what is a region of a country what is a country. Some subjects saved documents that discussed regions with declining birth rates within countries, but that did not necessarily imply that the country as a whole had a declining birth rate. Some subjects also saved documents that predicted that particular countries would in the future have a declining birth rate. Topic 446 ("In what countries have tourists been subjects to acts of violence causing bodily harm or death") is another example. Some subjects saved documents that mentioned countries in which attacks on tourists had occurred but which were judged not relevant (no aspects present) by the assessors because the documents did not explicitly indicate that bodily harm or death had in fact occurred. To reduce the judgement mismatch between subjects and assessors, more detailed or specific description of instances may be needed.

Subject satisfaction

User satisfaction is also an important measure of human-computer interfaces. A user-satisfaction questionnaire, adapted from [4], was used to assess each user's satisfaction with the

Topic	Number of documents								
	5	10	15	20	50	100	150	200	300
408	0.250	0.292	0.333	0.375	0.542	0.750	0.750	0.750	0.792
414	0.667	0.833	0.833	1.000	1.000	1.000	1.000	1.000	1.000
428	0.192	0.308	0.346	0.423	0.538	0.731	0.769	0.808	0.846
431	0.325	0.450	0.500	0.500	0.750	0.975	1.000	1.000	1.000
446	0.125	0.125	0.125	0.125	0.250	0.438	0.438	0.438	0.438
AV:	0.312	0.402	0.428	0.485	0.616	0.779	0.791	0.799	0.815

Table 11: Aspectual recall for the candidate document pool of the 300 highest-ranked documents

	408	414	428	431	446
Total aspects in all documents	24	12	26	40	16
Total documents containing aspects	71	16	40	67	58
Aspects in candidate documents	19	12	22	40	7
Candidate documents containing aspects	56	16	30	66	21

Table 12: Aspect coverage for each topic, as judged by the NIST assessors.

two interfaces. The questionnaire focused on subjects' satisfaction with the presentation format, the delivered data, an interface's ease-of-use, and the time available for the topics. Members of one group of 12 subjects were required to fill in this questionnaire after completing each session of three topics with an interface. Subjects were asked to respond to the questions using a five point Likert-type scale, where 1 = almost never; 2 = some of the time; 3 = about half of the time; 4 = most of the time; and 5 = almost always.

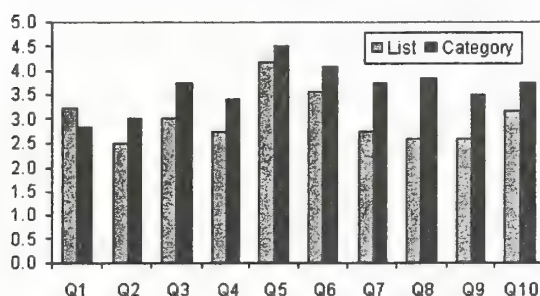


Figure 5: The results from subjects' satisfaction questionnaire (Q1: Too much information, Q2: Precise instances, Q3: Sufficient instances, Q4: Enough search time, Q5: Easy to use, Q6: User friendly, Q7: Clear organization, Q8: Useful format, Q9: Organization what is needed, Q10: Satisfaction with the interface.)

The results from the questionnaires are shown in Figure 5. The X-axis shows the questions asked, and the Y-axis the mean score for each question. Q1- Q3 shows subjects' satisfaction with the displayed contents; Q3 the time available; Q5-Q6 the ease of use; Q7-Q9 the way the data was organized; and Q10 overall satisfaction with the interface. Note that for Q1, a lower score indicates greater satisfaction. From Figure 5 we can see that, for all questions, the satisfaction scores for the classification-based interface are higher than the ranked list interface. This difference is statistically significant ($p < 0.001$, paired, one tail t- test).

Figure 5 also suggests that the organization of retrieved

data may influence the subjects' perception of it. Although both interfaces offered the same amount of information, albeit differently organized, subjects nonetheless felt that the ranked list interface showed too much information, and felt able to find neither enough information nor sufficiently precise information to answer the topics. Given that subjects saved approximately the same number of facts with each interface, this indicates a strong discrepancy between subjects' preferences and their performance.

Although most comments from the post-experiment questionnaire regarding the classification-based interface were positive, offering suggestions for further improvement, some negative ones were made. Subjects wanted the set of categories to be derived from more than one keyword for some topics; for example, topic 408 includes the phrase "tropical storm" which is a more logical basis for categorization than either "tropical" or "storm" in the context of the topic. Subjects also disliked only having one chance to select the classification axis; when subjects later decided they had made the wrong choice, there was no mechanism for them to undo that decision. We had recognized that such a mechanism would be a desirable feature beforehand, but chose not to include it in this preliminary experiment in order to focus on evaluating dynamic classification. Our main object has been to determine whether a classification-based interface can help the subject to find more facts, under the assumption that a subject can select a right set of categories. This assumption may not have been a valid one, as previously noted.

Another problem observed was that some classifications resulted in too many categories being created (over 500 in one case). This could be addressed by imposing a maximum for the number of categories that may be formed; an appropriate value for this threshold would need to be determined. An alternative is to use some form of hierarchical or grouped super-categorization.

A general problem we experienced is the shortcomings of using WordNet as the source for hyponyms. Not surprisingly, WordNet was not always able to supply hyponyms appropriate to the context of the topics. Additionally, the issue of sense-selection applies here, as in all natural-language dependent systems.

	Read List		Saved List	
	category Mean (SD)	list Mean (SD)	category Mean (SD)	list Mean (SD)
Documents read/saved	18.7(10.5)	16.0(7.2)	6.35(4.06)	6.30(2.6)
Aspectual Precision t-test	0.44(0.26)	0.52(0.26)	0.72(0.26)	0.75(0.24)
	< 0.04 (<i>sig.</i>)		< 0.26 (<i>not sig.</i>)	
Aspectual Recall t-test	0.36(0.23)	0.41(0.20)	0.29(0.21)	0.32(0.19)
	< 0.08 (<i>not sig.</i>)		< 0.19 (<i>not sig.</i>)	

Table 13: Comparison of performance between category and list - for five topics

4.5 Conclusion

We evaluated our preliminary hypothesis of organizing retrieved documents according to the intentions behind a question. We are glad to see that subjects are significantly more satisfied with the classification based interface. Although our current experiment result did not show a significant benefit in terms of aspectual recall by using classification based interface, we still believe it has a potential and could be improved. The further improvements may include:

- Utilizing both the phrases and the single words as the basis for classification.
- Allowing users to select more than one term or phrase as the basis for classification.
- Allowing users to remove or modify individual categories; this may provide a user-driven way around problems of sense-selection.
- Providing a better description of each potential classification and how it was derived.
- Developing a better ranking for categories.

5 Spoken Document Retrieval Track

Two runs were submitted for this year's Quasi-SDR runs. The word-based documents were first translated to phonemes using a text-to-phoneme algorithm. We assumed that there is a certain level of word recognition error for each type of transcription. Given this, we utilised a passage retrieval technique to perform the retrieval.

5.1 Text-to-phoneme Algorithm

The words were translated to phoneme using a modified version of the text-to-speech translation algorithm given by Carney [2]. The original algorithm was modified to produce American pronunciation as close as possible to those given by CMU [3]. The advantage of this approach is the implicit handling of unknown words. This will not be a problem if there is a translation dictionary containing all the words in the collection. As a test, the reference and baseline transcriptions were translated to phonemes using the CMU pronunciation dictionary [3]. There were approximately 18,000 and 300 unknown words in the reference and baseline transcriptions respectively. The reference collection has approximately 34,500 unique words while the baseline collection has about 19,000 unique words. This may mean that a lot of unknown words were not being recognised by the automatic recognition system. A problem of this algorithmic approach is the inconsistencies in the generation of some of the pronunciations.

Len	1	2	3	4	5	6
# of Words	11	32	45	51	55	46
Len	7	8	9	10	11	12
# of Words	5	28	16	12	4	4

Table 14: Distribution of unique query words for different phoneme lengths.

5.2 Matching Algorithm

The algorithm described here is based around the notion of a passage for each indexing feature, whether that feature is a word or a sequence of phonemes. In this instance, a passage is defined for a query term which is a sequence of phonemes translated from a query word. Potential passages are found within each document for each query term by way of approximate matching. Later, the retrieved passages for each query term is weighted according to *tf.idf* scores and combined for each document before similarity scores are calculated.

The following subsections describe each individual components.

Window Size

The window size determines the error margin allowed for the transcriptions. If the word error rate (or WER) of a recogniser is small, then it is possible to have a small window and yet find all the phoneme sequences with a high degree of match. For higher WER, a larger window may be required to find all the phoneme sequences describing the required word, although this may lead to higher false matches.

From the translation of the query words to phoneme sequences, we found that phoneme sequences range from 1 phoneme to 12 phonemes (See Table 14 for further details). Most words were between 4 to 6 phonemes long. Here, the assumption made is that query terms of medium length required a larger error margin than longer query terms.

For each term in the query, we have to determine a suitable window size (or passage length) for matching documents. If the number of phonemes in the query term is less than 8, length of the matching window is set to number of phonemes times 1.5. For query terms with more than 8 phonemes, window size is set to number of phonemes plus 4.

Given that short query terms are likely to have many matches, both false and correct ones, those that are shorter than three phonemes are discarded and not used in the matching and ranking processes. This acts as a form of stopping.

Phoneme-based Matching

The scores for matching sequences for each query term is calculated. First, the size of a passage is determined by the win-

dow size described previously. Therefore for each passage, its size is a constant for that particular query term or phoneme sequences. At this stage, we need to maximise the score for the longer sequences matched. Then the score is penalised if the sequences matched in the passages are not in-order.

Passages in documents are scored as follows. Locate all possible *exact* phoneme matches between a query term t and passage p . Sort them according to the longest sequence matched in decreasing order. Using the longest sequence as a starting point, determine as many phoneme sequence matches for each passage as possible. The passage score is the sum of square of non-overlapping phoneme matching sequences:

$$Score_1(p, t) = \sum_{m \in M} [len(m)]^2 \quad (11)$$

where M is a set of phoneme sequence matches found between t and p . $len(m)$ is the length of m expressed in the number of phonemes. By taking the $len(m)^2$, we will always favour the longest sequence matched in any passage.

Equation 11 does not make a distinction between passages that have phoneme sequences in the same order as in the query term from those passages that are not. For example, the query HAIR has the phoneme sequence "HH EY R", while GREYHOUND is translated as "G R EY HH AW N D". The false matching of "HH" and "EY" in GREYHOUND will have the same score as other words containing the same sequence in the right order. As a result, we attempt to "penalise" those passages where disjoint phoneme sequence matches do not have the same order as in the query term. We devised a simple heuristic that attempts to find all matching sequences in passages that are not in order and use their sum as the factor to penalise the passage score:

$$Penalty(p, t) = \sum_{u_m \in U_M} [len(t) - len(u_m)]$$

where U_M is a set of phoneme sequence matches between t and p that are found to be out of order with respect to t . Therefore the refined passage score becomes:

$$Score_2(p, t) = Score_1(p, t) - Penalty(p, t) \quad (12)$$

We note that other, possibly better techniques, could be used to distinguish such passages. This is an ongoing work for this approach.

A final modification to $Score_2(p, t)$ is required so passages selected for individual query terms can be combined in order to arrive at the overall score of documents. Passage scores are normalised with respect to the number of phonemes in the query term:

$$Score_3(p, t) = Score_2(p, t) / len(t) \quad (13)$$

This also ensures that short query sequences aren't penalised too heavily although longer sequences are usually found with higher accuracy. At the end of the matching, we output 1000 best passages with the highest score (that is, $Score_3(p, t)$). Note that this scoring mechanism permits multiple passages to be matched within a document.

Individual Term-passages Weighing

Selecting passages is not much use if documents need to be ranked. The process described in the previous section could be thought of as a means of word-spotting. The aim here

however, is to rank documents. Therefore, taking the approximate word matches as produced in the previous section, we attempt to weigh them in documents.

The idea is to assume that passages with high scores are likely to correspond to the "true" word matches in the documents. By "true" we mean that the actual word appears in the document but was possibly misrecognised.

Passage scores, as computed above, can be considered as an indicator of approximate word matches in documents. Therefore, passage scores are used to accumulate in-document frequencies for query terms as follows:

$$f_{d,t} = \sum_{p \in P} \frac{Score_3(p, t)}{Score_{max}(t)}$$

where P is the set of passages matching the query term t in document d . Passage scores are normalised with respect to the maximum score for the query term t ($Score_{max}(t)$). Passages with low scores are discarded because they are likely to be false matches.

Similarly, while computing $f_{d,t}$, we estimate document frequency for query term t (f_t) as the number of documents in which $f_{d,t}$ is higher than zero. (Note: this may be undesirable because the query terms will not be discriminated well because most of them will have high f_t values.)

Given the weights of query terms in documents, a final step is to compute the similarity score of query q to document d . This can be done by using standard tf.idf weighting [10] or more modern weighting approach such as Okapi that we have described in Section 1.2 (see Equation 1). (Note that in our case the matches of query terms in documents are not exact due to our approximate matching algorithm.)

Results

The retrieval effectiveness of our official run (mds-base) is low. This is due to mainly two reasons. First, term weights in documents were computed using the standard tf.idf formulation [10], and no document length normalisation was used. In our post-TREC runs we added Okapi measure with document length normalisation as an alternative to tf.idf weighting. The official run that used the base transcription (BASE) is marked as mds-base in Table 15. In addition there is mds-ref run which used identical approach to mds-base except the collection was a manual transcription (REF).

Besides standard tf.idf weighting there is room to improve in the way document frequencies are estimated and the document length component be used in computing similarities for documents. Our basic approach estimates document frequencies for terms (or f_t) by assuming that if there is at least one "partial" match of the query term in document, this document is counted (that is when $Score_3(p, t)$ is greater than zero for passage p in a document). It turns out that most query terms end up with high f_t values because of the approximate matches on documents. The skewed distribution of f_t towards high values leads to poor discrimination of terms when similarities between documents and queries are computed [10].

There is no obvious way of defining document frequencies in the context of phoneme based retrieval because there is no notion of words in documents. However, we define an upper bound for retrieval using the basic window-based ranking as described earlier by extracting document frequencies from REF collection. (Note that in real situation this information would not be available.) The retrieval effectiveness for this run is marked as ft_{REF} in Table 15.

Run	p@5	p@30	AvgP	AvgP(Revised)	Recall
<i>Text-based runs (REF collection)</i>					
q-unstopped	0.5320	0.2947	0.3857	-	1518
q-stopped	0.5200	0.2953	0.3781	-	1558
<i>Text-based runs (BASE collection)</i>					
q-unstopped	0.5000	0.2740	0.3197	-	1381
q-stopped	0.4880	0.2747	0.3169	-	1464
<i>REF (manual) converted to phonemes</i>					
mds-ref	0.2920	0.1647	0.1740	0.1775	1398
<i>Base (word-recogniser) converted to phonemes</i>					
mds-base	0.2320	0.1293	0.1323	0.1350	1251
<i>ft_{REF}</i>	0.3680	0.2107	0.2119	-	1305
<i>ft_{REF},Wd_{REF}</i>	0.4040	0.2173	0.2445	-	1275
<i>ft_{WSJ},Wd_{REF}</i>	0.3960	0.2280	0.2752	-	1347
<i>ft_{TREC},Wd_{REF}</i>	0.3880	0.2340	0.2811	-	1349

Table 15: Retrieval results for reference and baseline transcripts using Cosine-based weighting.

An alternative for approximating document frequencies is to use an auxiliary collection which is not related to the speech collection. In this experiments we used WSJ data from disk 1 and 2 of TREC and TREC-7 (also used in TREC-8). The retrieval effectiveness for this run is marked as *ft_{WSJ}* and *ft_{TREC}* in Table 15 respectively.

Another shortcoming of our official run is that there was no document length normalisation; longer documents are expected to be favoured because they are more likely to have more matches of query terms than shorter documents. However, there is no clear way of incorporating document length (or *Wd* for document *d*) that is expressed in terms of phones into the standard vector space model. Under superficial circumstances we assume that we have access to document lengths as produced by REF collection. These are used in conjunction with *ft_{REF}* and is shown as *Wd_{REF}* in Table 15.

As it can be seen, both document frequencies and document lengths improve effectiveness.

Table 16 illustrates our results using Okapi-based weightings. Here, the number of phonemes were used to determine the the document length *Wd_{phn}* instead of the estimated document lengths in terms of words as used in the Cosine-based weighting. Table 17 gives use the results for the base transcripts in phonemes where the queries were stopped prior to being translated to phoneme sequences. At this moment, we are unable to draw any conclusions from these results.

We also try to eliminate passages that had at least one matching sequence that is not in order of the query term. This is marked as "*ft_{REF},Wd_{phn},force_order*" and had no impact on retrieval. However, this might not be right if a good sequence happens to align with another sequence matching by chance in the right order. Such possible good passages would not be discounted in this case.

5.3 Conclusion for Quasi-SDR

This year we attempted a passage-based technique to perform retrieval using phoneme sequences. Documents and queries were first translated to phonemes using a rule-based text-to-speech algorithm. A passage was created for each query term and approximate matches were computed within each document. These passages were combined using either cosine or Okapi-based weighting scores for each document before similarity was computed for each query.

Table 15 illustrates an important point. Although the phoneme based retrieval of speech documents using word-

based transcripts is not as effective as when words are used, the approach is more general than text-based retrieval. Retrieval of word-based transcripts is effective as long as two conditions are met: word recognition accuracy is high and a single language is used. For noisy word-based speech recognition, accuracy can be as low as 40%. On the other hand, phoneme based retrieval using ngrams and approximate matching works well under good conditions in comparison to word-based approach but will be superior in cases when poor recognition in which our data is not a good representation. Approximate matching methods like Wechsler [14] and Ng [8] are similar to this technique but different because recognition information like confusion matrices are required to determine the approximate matches. Note however, that recent attempts to recover from poor transcripts from a word-based recogniser was shown to be effective [12].

More analysis and experimentations are required to further test the assumptions made and to improve on our approximate matching strategies. Variables to investigate include determining an optimal window size, calculating the scores in the approximate matching process and the weighting algorithms for similarity computations.

Acknowledgements

The speech experiments were conducted while on leave at Sharp Laboratories of Europe Ltd. Thanks to Dr. Victor Poznański for discussions and encouragement. Also to the IT group for providing the resources.

References

- [1] N. J. Belkin, P. G. Marchetti, and C. Cool. Braque: Design of an interface to support user interaction in information retrieval. *Information Processing and Management*, 29(3):325–344, 1993.
- [2] E. Carney. *A Survey of English Spelling*. Routledge, London, 1994.
- [3] Carnegie Mellon University Pronouncing Dictionary. Cmudict.0.4. Available from <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>, 1995.
- [4] William J. Doll and Gholamreza Torkzadeh. The measurement of end-user computing satisfaction. *MIS Quarterly*, pages 259–274, June 1988.

Run	p@5	p@30	AvgP	Recall
<i>Text-based runs (REF collection)</i>				
q-unstopped	0.6240	0.3307	0.4399	1537
q-stopped	0.6160	0.3300	0.4389	1568
<i>Text-based runs (BASE collection)</i>				
q-unstopped	0.6000	0.3073	0.3888	1451
q-stopped	0.5840	0.3047	0.3877	1485
<i>Base (word-recogniser) converted to phonemes</i>				
base	0.2640	0.1493	0.1553	1248
ft_{REF}	0.4240	0.2320	0.2438	1329
ft_{REF}, Wd_{phn}	0.5000	0.2567	0.3045	1316
ft_{WSJ}, Wd_{phn}	0.4160	0.2187	0.2780	1299
ft_{TREC}, Wd_{phn}	0.3560	0.2033	0.2482	1185
$ft_{REF}, Wd_{phn}, force_order$	0.5000	0.2553	0.3041	1317

Table 16: Retrieval results for reference and baseline transcripts using Okapi-based weighting.

Run	p@5	p@30	AvgP	Recall
<i>Base (word-recogniser) converted to phonemes</i>				
base-stopped queries	0.4160	0.2307	0.2376	1337
ft_{REF}	0.4200	0.2380	0.2544	1329
ft_{REF}, Wd_{phn}	0.5040	0.2600	0.3075	1317
ft_{WSJ}, Wd_{phn}	0.4160	0.2187	0.2781	1308
ft_{TREC}, Wd_{phn}	0.3640	0.2067	0.2543	1250
$ft_{REF}, Wd_{phn}, force_order$	0.5040	0.2587	0.3070	1318

Table 17: Same as Table 16 but for stopped queries.

- [5] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge Massachusetts, 1998.
- [6] M. Fuller, M. Kaszkiel, D. Kim, C. Ng, J. Robertson, R. Willinson, M. Wu, and J. Zobel. TREC7 ad hoc, speech, and interactive tracks at MDS/CSIRO. In *Proceeding of Seventh Text Retrieval Conference (TREC-7)*, November 1998.
- [7] J.B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computation*, 11(1-2):22-31, 1968.
- [8] K. Ng. Towards robust methods for spoken document retrieval. In *Proceedings of International Conference on Spoken Language Processing*, volume 3, pages 939 - 942, Dec 1998.
- [9] J.J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall, Inc., 1971.
- [10] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, USA, 1983.
- [11] A. Singhal, C. Buckley, and M. Mitra. Pivoted Document Length Normalisation. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996.
- [12] A. Singhal and F. Pereira. Document expansion for speech retrieval. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 34-41, 1999.
- [13] S. Walker, S. Robertson, M Boughanem, G. Jones, and K. S. Jones. Okapi at TREC-6 Automatic ad hoc, VLC, routing, filtering and QSDR. In *Proceedings of the Sixth Text REtrieval Conference. (TREC-6)*, 1997.
- [14] M. Wechsler, E. Munteanu, and P. Schauble. New techniques for open-vocabulary spoken document retrieval. In *Proceedings of the 21st ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 20 - 27, Melbourne, Australia, Aug 1998.
- [15] Ross Wilkinson and Michael Fuller. Integrated information access via structure. In Maristella Agosti and Alan Smeaton, editors, *Information Retrieval and Hypertext*, pages 257-271. Kluwer Academic Publishers, 1996.
- [16] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes: Compressing and indexing documents and images*. Van Nostrand Reinhold, 1994.

Relevance Feedback *versus* Local Context Analysis as Term Suggestion Devices: Rutgers' TREC-8 Interactive Track Experience"

N.J. Belkin, C. Cool*, J. Head, J. Jeng, D. Kelly, S. Lin,
L. Lobash, S.Y. Park, P. Savage-Knepshield**, C. Sikora**

School of Communication, Information & Library Studies, Rutgers University

*Graduate School of Library and Information Studies, Queens College, CUNY

**Lucent Technologies, Holmdel NJ

Abstract

Query formulation and reformulation is recognized as one of the most difficult tasks that users in information retrieval systems are asked to perform. This study investigated the use of two different techniques for supporting query reformulation in interactive information retrieval: *relevance feedback* and *Local Context Analysis*, both implemented as term-suggestion devices. The former represents techniques which offer user control and understanding of term suggestion; the latter represents techniques which require relatively little user effort. Using the TREC-8 Interactive Track task and experimental protocol, we found that although there were no significant differences between two systems implementing these techniques in terms of user preference and performance in the task, subjects using the Local Context Analysis system had significantly fewer user-defined query terms than those in the relevance feedback system. We conclude that term suggestion without user guidance/control is the better of the two methods tested, for this task, since it required less effort for the same level of performance. We also found that both number of documents saved and number of instances identified by subjects were significantly correlated with the criterion measures of instance recall and precision, and conclude that this suggests that it is not necessary to rely on external evaluators for measurement of performance of interactive information retrieval in the instance identification task.

1. Introduction

Continuing our program of studying different methods of query expansion in interactive information retrieval (IR), this year our group investigated the effects of varying methods of term suggestion for user-controlled query expansion. The two methods that we compared were user control over suggested terms, implemented as positive relevance feedback (RF), versus magical term suggestion, implemented as a form of Local Context Analysis (LCA). We chose these two since they exemplify two polar methods for supporting interactive query expansion. The effects that we were most interested in were in terms of user preference, usability (as indicated by effort), and effectiveness in task performance.

Previous investigations by us (e.g. Koenemann, 1996; Park, 1999) and others (e.g. Shneiderman, 1998) have indicated that users in IR and similar systems generally prefer to have some measure of control on what the system does for them. This has often been in conjunction with an expressed desire to understand how the system has come to its suggestions/actions. These kinds of results led us to conclude that in interactive IR RF is best implemented as a term-suggestion device, rather than as an automatic query expansion device. In TREC-8, we decided to investigate the issues of control and understanding of system operation in more detail, by comparing a system in which users could control (and therefore presumably understand) where system-suggested terms came from (using RF with positive relevance judgments), with one in which suggested terms appeared as if by magic (*i.e.* LCA). Based on the previous work in this area, we hypothesized that user-controlled term suggestion would be preferred to system-controlled term suggestion.

As do others, we believe that a more usable system is a better system, and further, that a good indicator of usability is the amount of effort (physical, cognitive) that a person has to expend in order to complete a given task. We hypothesized that system-controlled term suggestion would require less effort on the part of the user than one which asked the user to make relevance judgments in order to get suggested terms. Such a difference is indicated by total time taken to perform the task, by the number of documents that a person looks at or reads, by the amount of use of various system features, and by the extent to which system-suggested terms are incorporated into the queries.

The TREC-8 Interactive Track task of instance identification is one which asks users to identify documents which

are about a number of *different* aspects of a general topic. Since RF is based on the idea of constructing an ever better (e.g. more specific) query, and since RF in interactive IR is typically based on a relatively small number of documents, it seems that RF term suggestion based only on positive relevance judgments is not well suited to this task. We can call such term suggestion *directed*. However, the terms identified by LCA for query expansion are based on a system-defined retrieved set of documents, as well as characteristics of the terms in the collection as a whole. Compared to RF, such term suggestion can be characterized as *diffuse*. We hypothesized that for the instance recall task, diffuse term suggestion would be more effective than directed term suggestion, and therefore that users would perform better in the LCA system than the RF system.

The standard measure of performance in the TREC-8 Interactive Track task is instance recall, defined as the proportion of instances of a topic that have been identified by the TREC judges, which have been identified by the searcher (as indicated by the documents the searcher has saved). Since the task that was set the searchers was to identify and save all of the instances of a topic, and since we are interested in developing evaluation measures for interactive IR that do not depend upon external relevance (and related) judgments, we also measured performance according to the number of documents saved, and the number of instances identified.

Thus, we suggest that although a term-suggestion feature based on RF might be preferred by users to one which is based on LCA, for reasons of control and understanding, the magical method will require less effort, and will lead to better performance in the instance identification task. It will not escape the reader's notice that these hypotheses seem, on the face of it, to be contradictory. That is, it does not follow naturally that a system which required less effort of the user to perform the user's task better, would not also be preferred. This situation provides another rationale for our investigations of these two conditions of term suggestion.

2. System Descriptions

There were two experimental IR systems used in this study. Both systems used Inquiry 3.1p1 with its default values for indexing and retrieval (cf. Callan, Croft & Harding, 1992). The sole difference between the two systems lies in the implementation of the term suggestion feature (this leads also to minor differences in the interfaces).

The first system, called INQ-RF, allowed users to make positive relevance judgments on documents. Inquiry's RF function was modified so that it displayed a list of terms for positively judged documents, rather than automatically expanding the query. As users made RF judgments about documents, the top n terms were presented in a term suggestion window. The number of terms displayed was determined by the formula:

$$n = 5i + 5$$

in which i is the number of judged documents, and n is no greater than 25. The term ranking algorithm was *rdfidf* (Haines and Croft, 1993), where *rdf* is the number of relevant documents in which the term appears, and *idf* is normalized inverse document frequency as used by Allen (1995) (cf. Belkin, *et al.*, 1999).

The second system, INQ-LCA, employed a slight modification of the technique called Local Context Analysis (LCA) (Xu and Croft, 1996) for term suggestion. LCA combines collection-wide identification of concepts, normally nouns and noun phrases, with co-occurrence analysis of those concepts with query terms in the top n passages retrieved by a query. The concepts are ranked according to a function of the frequencies of occurrence of the query terms and co-occurring concepts in the retrieved passages, and the inverse passage frequencies for the entire collection of those terms and concepts. The top m ranked concepts are then used for query expansion. In our version of LCA, these m ($m=25$, to match the RF condition) concepts were displayed in a term suggestion window, after each new query. Based on an experiment using the TREC-7 ad hoc task in which we compared performance of automatic LCA query expansion using different values of n and different definitions of passages (with m constant at 25), *passage* in our study was defined as the whole document, and n was set to 10.¹

Both systems used the same basic interface, developed at Rutgers, which offers the functions and features described below. Appendix A is a screen shot of the INQ-RF interface. The INQ-LCA interface was identical, except that there were no check boxes to indicate positively judged documents, and no **Clear Good Docs** button.

1 David Harper has pointed out to us that there is an inconsistency in our using the ad hoc task in these experiments, since that task is quite different from the instance recall task, especially in ways that might be relevant to choice of number of passages to be examined.

Suggested terms could be added to the existing query at the user's discretion, which is the same for both systems.

- Query terms window – used to input a free-form query, with minimal structure (phrases and negative terms).
- Results Summary window – displayed the titles of ten documents and provided check boxes for marking documents as good (only in the case of INQ-RF) and saved.
- Document window – displayed text of a selected document.
- Pop-up Instance Labeling window – used to label saved documents according to the "instances" that they represented.
- Documents Saved window – listed the saved document's title and its associated instance label(s).
- Good Terms to Add window – displayed suggested terms which could be added to the query by clicking on them.
- Search Button – used to retrieve a list of documents.
- Clear Query button – used to remove all terms in the query terms window.
- Clear Good Documents (only in the case of INQ-RF) – used to "unmark" previously marked good documents.
- Show Next Keyword, Show Best Passage, Show Next, and Show Prev buttons – used to quickly navigate through the full text of a document.
- Exit button – used to end a search session.

Both systems ran on a SUN Ultra 140 with 64MG memory and 9GB disk under Solaris 2.5.1 with a 17" color monitor.

3. Description of Study

A total of 36 volunteer searchers, recruited from the Rutgers community, participated in this project. Most (89%) of the subjects were full time students, who received compensation in the form of extra course credit for their participation. None had taken part in previous TREC studies. Each subject conducted six searches in accordance with the TREC-8 Interactive Track experimental guidelines. Subjects conducted three searches in both the RU-INQ and RU-LCA systems. We used a Latin square design where six topics were randomized and rotated completely so that each topic may appear only once in each row and only once in each column. The same set of topics was rotated again with a different system order, in order to allow a direct comparison between two different systems. Three different combinations of topic order and system order were used allowing us to run experiments with 36 subjects.

On arrival, the subjects read and signed a consent form explaining their rights and potential risks associated with participation in the experiment. Then they completed a demographic questionnaire that gathered background information and probed their previous searching experience. Next, they received a hands-on tutorial for the first system, describing the various features of that system. After completing the tutorial, subjects were given a general task description and specific instructions for the current search topic. They were allotted 20 minutes to complete each search. As they searched, they labeled instances of topic as they identified them and saved documents. During the sessions participants were asked to continuously "think aloud." A videotape recorded the computer monitor during their searches and also captured their "thinking aloud" utterances. The entire search interaction was logged.

After conducting each search, subjects answered several questions about their familiarity with the search topic, experiences with the searching task, their satisfaction with the search result, and satisfaction with the amount of time allotted for the search. After completing three searches for the first system, subjects answered several questions about the system in general. After a short break, the subjects were given a tutorial for the second system, searched another three topics, completed a post-search questionnaire for each topic, and a post-system questionnaire. After completing all six searches, the subjects completed an exit interview. The entire session took between 3 and 3 and one-half hours.

As mentioned above, an overwhelming majority (89%) of the subjects were students, and 83% were female. The average age of these searchers was 24 years old. Seventy-five percent of the subjects held, or expected to receive, a bachelor's degree at the time of the experiment. Nineteen percent had, or expected to receive, an MLS. On average, these searchers had been doing online searching for just over three years ($M = 3.48$).

We asked a series of questions about the background experiences of our volunteer searchers, using a 5 point scale, wherein 1= no experience and 5= a great deal. Overall, the searchers were quite familiar with the use of GUIs ($M=4.19$), and with the WWW search engines ($M=4.06$). A majority reported having had some experience with library OPACs ($M=3.22$), and with searching on CD ROM systems ($M=3.0$). In light of this, it is not surprising that on average our searchers reported conducting searches greater than twice a month.

Of note is that experience searching on commercial online systems in general was reported to be fairly low for our subjects ($M=2.19$) and experience searching on systems other than the web was markedly low ($M=1.19$). On a final note, the searchers in our study tended to say that they enjoyed doing information searches ($M=3.56$) as measured by the 5 point scale wherein 1= strongly disagree and 5= strongly agree.

4. Results: Descriptive Statistics

The interaction between the searcher and the system was similar for the two systems as can be seen by the means presented in Table 1. The mean number of documents retrieved during a search topic was almost identical for LCA ($M=653.59$) and RF ($M=655.49$). The number of iterations (queries) in a search was roughly the same for the two systems (LCA $M=5.93$, RF $M=5.76$). Consistent with the data for documents retrieved, mean number of unique titles displayed for a search topic was also similar for LCA and RF (123.53 and 128.81, respectively). The searchers viewed the full text of almost 20% of the unique document titles displayed in each system (LCA $M=21.88$ and RF $M=25.41$). The similarity between the systems, in terms of interaction, was also demonstrated by the number of instances identified and documents saved. The mean number of instances identified by a searcher for a particular topic was 9.36 for LCA and 9.75 for RF. The mean cumulative number of documents saved was 8.66 for LCA and 8.69 for RF. Given that multiple instances could be found in one document, more instances were identified than documents saved in both systems. System errors generally did not occur in either system (LCA $M=.01$ and RF $M=.01$).

	LCA M (SD)	RF M (SD)
Number of documents retrieved:	653.59 (383.88)	655.49 (533.83)
Number of unique titles displayed:	123.53 (60.77)	128.81 (69.86)
Number of unique full-texts viewed:	21.88 (10.82)	25.41 (18.94)
Number of instances identified:	9.36 (4.79)	9.75 (5.69)
Number of documents saved:	8.66 (4.33)	8.69 (4.79)
Number of system errors:	.01 (.30)	.01 (.30)
Number of documents identified as "good":	NA	2.85 (3.40)
Number of times suggested term list was cleared:	NA	.21 (1.01)
Number of times the good mark was removed:	NA	.74 (1.88)
Number of times the query window was cleared:	.95 (1.54)	.87 (1.62)
Number of times the save mark was removed:	.17 (.50)	.18 (.41)
Number of times paging-style scrolling was used:	22.26 (17.48)	25.44 (19.49)
Number of times dragging-style scrolling was used:	1.36 (2.32)	1.80 (3.58)
Number of suggested terms:	126.61 (83.82)	113.03 (148.93)
Number of <i>unique</i> suggested terms:	62.73 (33.59)	22.81 (24.62)
Number of unique terms used in the query:	10.0 (5.60)	8.91 (5.71)
Number of suggested terms selected to use in the queries:	4.41 (5.23)	1.87 (2.65)

Table 1: Means and standard deviations associated with system interaction and feature use.
Note: For LCA and RF, $n = 108$. Each mean is based on one search topic session.

Feature use is another aspect of system interaction. Feature use was fairly comparable in the two systems with the exception of the additional actions required to obtain suggested terms in the RF system. In the RF system, the average number of documents that were identified as relevant and used to generate suggested terms was 2.85 per search topic. Searchers generally did not use the features to clear the suggested terms list or uncheck a document as relevant ($M = .21$ and $M = .74$, respectively). On average for both systems, searchers cleared the query window no more than one time per search topic (LCA $M = .95$ and RF $M = .87$). Searchers generally did not change their mind and 'unsaved' a document in either system (LCA $M = .17$ and RF $M = .18$). Across the two systems, similar frequency of use within a search session was found for paging-style scrolling (LCA $M = 22.26$ and RF $M = 25.44$) and dragging-style scrolling (LCA $M = 1.36$ and RF $M = 1.80$). Overall, these descriptive statistics demonstrate similar interactions for the two systems.

When looking specifically at query formulation and term suggestion, interaction differences between the two systems emerge. Although the total number of suggested terms was similar for LCA and RF ($M = 126.61$ and $M = 113.03$, respectively), the number of unique suggested terms provided by LCA and RF differed substantially ($M = 62.73$ and $M = 22.81$, respectively). Additionally, the total number of unique terms used in the query by the searcher was similar for LCA and RF ($M = 10.0$ and $M = 8.91$, respectively). However, the average number of suggested terms that the user selected to use in their query was very different (LCA $M = 4.41$ and RF $M = 1.87$). The strongest contrasts in the systems beyond those associated with the use of relevance feedback, are the differences in the number of unique terms suggested by the systems and the number of those selected by the searcher.

5. Results: Preference, Effort, Performance

In this section, we present the results of our study with respect to the three hypotheses which motivated it. Under each hypothesis, we show the results on the relevant measures, and indicate whether the hypothesis is supported or rejected according to those results.

5.1 Hypothesis 1: User-control (RF) will be preferred to system-control (LCA)

System preference was measured by subjective response to the following question: "Which of the systems did you like best overall?" System preference was distributed roughly evenly across the RF (39%), LCA (31%) and No Difference (31%) categories. This measure was not significantly related to system order nor to performance, but there was a relationship between preference and *perceived* system effectiveness.

Perceived system effectiveness was measured by response to questions which asked subjects about the extent to which they used each term suggestion feature to modify their searches, the extent to which they found the terms that were suggested by each of the systems useful and the extent to which the term suggestion feature improved their ability to identify different aspects of the topics in each of the systems. Each of these questions was measured on a 5-point Likert scale, where 1=not at all; 3=somewhat; and 5=a great deal. A factor analysis was performed in order to create an index variable of perceived system effectiveness for each system. One factor emerged for each of the systems that included all of the measures. The reliability coefficients for each of the system factors, Perceived System Effectiveness of LCA and Perceived System Effectiveness of RF, were high (LCA $\alpha = .849$, RF $\alpha = .777$). The mean Perceived System Effectiveness for LCA was 3.02 ($SD = 1.11$). For RF, the mean Perceived System Effectiveness was 2.94 ($SD = 1.03$). T-tests indicate that there is a significant difference in the Perceived System Effectiveness between subjects who prefer LCA [$t(214) = 2.9$, $p < .01$] and those who prefer RF [$t(214) = -2.85$, $p < .01$]. Those who preferred LCA scored significantly higher on Perceived System Effectiveness for LCA. Those who preferred RF scored significantly higher on Perceived System Effectiveness for RF.

In sum, as might be expected, subjects preferred the system that they perceived to be more effective. Their perceptions, however, were not related to objective performance measures.

Based on the relatively even distribution of system preference, hypothesis 1 is rejected.

5.2 Hypothesis 2: LCA will require less effort than RF

There was little difference in subjective responses to questions intended to measure effort on the two systems. When asked which system they found easier to *learn* to use, seventy-five percent of subjects indicated that there was 'no difference' between the two systems. The remainder of the subjects were closely split between

preferences for the two systems (LCA = 14% and RF = 11%). When asked which system was simply easier to use, fifty percent of subjects expressed no preference for one system over the other. The other fifty percent were again closely divided in their preferences between the two systems (LCA = 22% and RF = 28%). When the question focused on the ease of using the systems' term suggestion feature, only twenty-five percent indicated no clear preference. Of those searchers who had a preference, LCA's term suggestion feature was indicated as preferred more often (LCA = 42% and RF = 33%). There was no system order effect on these results.

The effort associated with interacting with the two systems was similar based on the use of features, number of iterations (queries), and the viewing of items (see Table 1 for the data on these measures). Neither page-style scrolling nor dragging-style scrolling yielded significant differences between the two systems [$t(214) = -1.26$, ns and $t(214) = -1.06$, ns, respectively]. The number of iterations (queries) in a search was roughly the same for the two systems (LCA $M=5.93$, RF $M=5.76$). The difference between the two systems was also insignificant for total number of documents viewed, total number of unique documents viewed, total number of titles displayed and total number of unique titles displayed [$t(214) = -1.71$, ns; $t(214) = -1.68$, ns; $t(214) = -1.14$, ns; $t(214) = -.59$, ns; respectively].

The total number of query terms used in a single query was roughly equivalent regardless of the system the user was using (LCA $M = 10.0$, RF $M = 8.91$). However, the way in which the terms were acquired for use in the query did vary across systems. The number of suggested query terms selected by the user was significantly higher when using the LCA system compared to the same users searching on the RF system, (LCA $M = 4.41$; RF $M = 1.87$; $t(214) = 4.50$, $p < .001$.) The number of user-defined terms entered into the query by the user, that is, those query terms *not* selected from the suggested terms list, was significantly higher for RF than LCA, (LCA $M = 5.59$; RF $M = 7.04$; $t(214) = 2.04$, $p < .05$). This suggests that in the RF system users spent more effort generating terms themselves, while in the LCA system users spent less effort thinking of terms and selected more terms from those provided.

Based on the measure of effort defined as the user's having to think of good query terms, hypothesis 2 is supported.

5.3 Hypothesis 3: LCA (diffuse term suggestion) will be more effective than RF (directed term suggestion)

Performance was measured by instance recall, number of instances identified and number of documents saved. The mean instance recall for the two systems was close (LCA $M = .24$, RF $M = .26$), as was the number of instances identified (LCA $M = 9.19$, RF $M = 9.56$). For number of documents saved, subjects' performance was almost identical (LCA $M = 8.48$, RF $M = 8.49$). These differences were all insignificant, which suggests that the effectiveness of the two systems is similar [$t(214) = -.69$, ns; $t(214) = .51$, ns; $t(214) = -.06$, ns; respectively]. There was no system order effect on these results. The means and standard deviations for each of the performance measures are displayed in Table 2.

	TOTAL M (SD)	LCA M (SD)	RF M (SD)
Instance Recall	.25 (.17)	.24 (.16)	.26 (.17)
Number of Instances Identified	9.38 (5.18)	9.19 (4.81)	9.56 (5.55)
Number of Documents Saved	8.49 (4.52)	8.48 (4.33)	8.49 (4.72)

Table 2. Means and Standard Deviations of Performance Measures

There was little difference in the subjective response to a question intended to measure effectiveness of the two systems. When asked which of the systems' terms they found more effective, forty-two percent of subjects indicated that RF suggested more helpful terms, thirty-three percent of subjects indicated that LCA suggested more helpful terms and twenty-five percent of subjects indicated that there was no difference in the helpfulness of the terms suggested by the two systems.

Correlations between the performance measures were computed in order to determine the relationship between measures that depend on external relevance judgments (instance recall) and measures that depend upon user

relevance judgments (number of instances identified and number of documents saved). All of the performance measures were significantly correlated, which suggests that number of instances identified and number of documents saved might be considered as alternative evaluation measures for interactive IR. These results are displayed in Table 3.

	Instance Recall	Number of Instances Identified	Number of Documents Saved
Instance Recall	1.00	–	–
Number of Instances Identified	.412*	1.00	–
Number of Documents Saved	.315*	.896*	1.00
*Correlation is significant at the .01 level			

Table 3. Correlation Matrix of Performance Measures

Since no significant differences in performance were found between LCA and RF, hypothesis 3 is rejected.

6. Discussion and Conclusions

Our first reaction to our results is HOORAY!! For the first time in the history of the TREC Interactive Track, it's been possible to realize a statistically significant difference between two treatments. Our second reaction is, however, somewhat more muted. Two out of three of our hypotheses were rejected, and the third was supported based on only one measure out of several. What should we make of these results?

To discuss the positive first, we found a significant difference in one measure of effort between the LCA and RF systems, while finding no significant differences in preference for the two systems, nor in the objective measures of performance in the task. We conclude from these results that the LCA system is better (i.e. more *usable*) for supporting the instance identification IR task than is the RF system. This finding also lends support to the idea that an IR system which suggests terms for query modification without user control is better than one which requires user control, that is that, as Croft (1995) suggested, users want magic. Less flippantly, we note that the mean number of unique terms that were suggested by LCA was about three times the number of unique terms suggested by RF. The ratio of suggested terms added to the queries in the two systems was also roughly 3:1. This suggests that having more terms to choose from leads to including more of those terms in the query.

How can we explain the result that user-controlled term suggestion was not preferred to "black box" term suggestion? We have three possible answers. One is based on general principles of interface design (e.g. Shneiderman, 1998), which suggest that as task complexity increases, the desire for, and effectiveness of user control, decreases. In the case of this study, we note that users had three tasks in common in the two systems: developing effective queries; deciding on whether a document should be saved; and labeling the instance associated with the document. However, in the RF system, the users had also to make decisions about which documents to mark good, and to consider the relationships between these documents and the terms that were suggested. Thus, there arose an explicit task associated with term-suggestion, which in and of itself was complex, and added a layer of complexity that didn't exist in the LCA system. Thus, the measure of control that was gained, was not worth the extra complexity it required.

Another possible explanation for non-preference of user-controlled term suggestion is that user control itself was not enough to overcome the effect of the other factors which might affect preference, in particular that of effort. Finally, it could also be the case that the difference between the two systems which we hypothesized to be quite significant, was not perceived as such by the subjects. This could be due to the novel task and situation in which the subjects found themselves, and the great similarity between the two systems on other dimensions.

In terms of performance, LCA turned out not to be better than RF, contrary to our hypothesis. For this result, we also have a potential explanation, based primarily on comments made by the subjects with respect to the nature of the terms suggested by LCA. The general opinion seems to have been that many of these terms were difficult to

understand: they were in languages other than English; they were proper nouns of unusual sorts; they were sometimes only numbers. It appears that some characteristics of the term-ranking algorithm used by this version of LCA (and perhaps the values we chose for the LCA parameters) favored quite rare co-occurrences (i.e. low document frequencies). A reasonably consistent comment in the exit interview was that if LCA presented "better terms", then it might have been preferred over RF, whose suggested terms were more understandable. This suggests experimenting with the term-selection algorithm used by "magic term suggestion", to see if "better" terms will lead to better task performance.

Overall, we conclude on the basis of this study, that magical term suggestion is likely to be a better mode of support for query modification than user-controlled term suggestion, in that control of term suggestion is less important to users of IR systems than is ease of use of a term suggestion feature.

We wish to make one final observation with respect to evaluation of interactive IR. In our study, the measures of performance applied to the Interactive Track task, instance recall and precision, were significantly correlated with the number of instances identified by the subjects, and with the number of documents saved by the subjects. This result suggests to us that it would be reasonable to evaluate performance in the instance identification task without having to depend upon external judgments, relying rather solely upon how "well" the subjects performed on the task that they were in fact set: identifying as many instances of a topic as possible in a given time period. But the other groups in the Interactive Track did not find significant correlations between these measures, which suggests that factors associated with our subjects, or our interface, might have led to our result. We nevertheless believe that this result could have important implications for the methodology of experimentation in interactive IR.

7. Acknowledgments

We would like to offer our heartfelt thanks to James Allan and Victor Lavrenko, both of the Center for Intelligent Information Retrieval at the University of Massachusetts, Amherst, for all of the help that they gave us in installing LCA at Rutgers.

8. References

- Allan, J. (1995) Relevance feedback with too much data. In *SIGIR '95*. Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, 337-343.
- Belkin, N.J., Perez Carballo, J., Cool, C., Kelly, D., Lin, S., Park, S.Y., Rieh, S.Y., Savage-Knepshield, P. & Sikora, C. (1999) Rutgers' TREC-7 interactive track experience. In *TREC-7*. Proceedings of the Seventh Text REtrieval Conference. Washington, D.C.: NIST, 275-283.
- Callan, J.P., Croft, W.B. & Harding, S.M. (1992) The INQUERY retrieval system. In *Dexa 3*, Proceedings of the Third International Conference on Database and Expert System Applications. Berlin: Springer Verlag, 78-83.
- Croft, W.B. (1995) What do people want from information retrieval (the top ten research issues for companies that sell and use IR systems). *D-Lib Magazine*, November 1995. <http://www.dlib.org/dlib/november95/11croft.html>
- Koenemann, J. (1996) *Relevance feedback: usage, usability, utility*. Ph.D. Dissertation, Department of Psychology, Rutgers University, New Brunswick, NJ.
- Park, S.Y. (1999) *Supporting interaction with distributed and heterogeneous information resources*. Ph.D. Dissertation, School of Communication, Information and Library Studies, Rutgers University, New Brunswick, NJ.
- Shneiderman, B. (1998) *Designing the user interface*, 3d edition. Reading, MA: Addison-Wesley.
- Xu, J. & Croft, W.B. (1996) Query expansion using local and global document analysis. In *SIGIR '96*. Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, 4-11.

Appendix: Screen Shot of System INQ-RF Interface

INQ System

Good Terms to Add

efficient
tyre
emission
clinton
air
clean
fuel
light
administrate
encourage

Edit
Query Terms

alternative energy sources for cars air

Good: 1
Saved: 1

+	1 . FT	17 AUG 94 /	Business and the Environment: Sparks fly over electric cars - Batter	-	Batter
+	2 . FT	13 OCT 94 /	World Trade News: Energy Research Projects - Contracts	-	Contracts
+	3 . FT	07 DEC 93 /	Survey of Energy Efficiency (2): On earth as in the atmosphere - En	-	En
+	4 . FT	27 JUL 94 /	Ford joins 'hybrid' electric car project	-	project
+	5 . FT	07 AUG 92 /	Phasing out is predicted for company car perk	-	perk
+	6 . FT	05 AUG 92 /	Business and the Environment: Fuel cells' chance to take charge	-	charge
+	7 . FT	17 NOV 92 /	Survey of Energy Efficiency (12): As free as the wind - There's not	-	not
+	8 . FT	02 NOV 92 /	Subsidy sought for waste projects	-	projects
+	9 . FT	29 MAY 92 /	House passes energy bill with oil-saving aim	-	aim
+	10 . FT	21 SEP 93 /	Letters to the Editor: Renewable energy industry makes sense	-	sense

FT 17 NOV 92 / Survey of **Energy** Efficiency (12): As free as the wind - There's nothing quixotic about renewables

FT924-7244
921117

THE range of renewable **energy** fossil fuels and nuclear power Wind, hydro, tidal, wave, geoth municipal wastes all have pass governments supporting research knowing how to allocate funds A study carried out last year for Support Unit (ETSU) at Harwell feasible to generate 25,000 MW electricity requirement - from today's meagre renewable gene the next two decades.

In practice it will not be possible to build up **alternative energy sources**

Instance Window

Please type some text that identifies the instance to which this document is relevant

wind

Documents saved
Saved: 1

+	FT	07 DEC 93 /	Survey of Energy Efficiency (2): On earth as in the atr	-	air
+				-	
+				-	
+				-	
+				-	

An Early DiscoWeb Prototype at TREC8

Brian D. Davison, Apostolos Gerasoulis,
Konstantinos Kleisouris, Yingfang Lu, Hyun-ju Seo,
Junyu Tian, Song Wang, Wei Wang, and Baohua Wu
Department of Computer Science
Rutgers, The State University of New Jersey
Piscataway, NJ 08855 USA

Recently the notion of popularity and its generalizations have been investigated as a possible alternative approach to text only analysis to rank web pages in search engines (e.g. [Kle98, BP98, CDR⁺98, CDDG⁺98, BH98, HHMN99] among others). We have built a research prototype that incorporates many link analysis algorithms from the literature and also new algorithms to investigate the impact of the popularity on the ranking of the search engines [DGK⁺99].

Our goal in the TREC8 competition was to investigate the quality of the results using the TREC data and in particular the large web track. Unfortunately we did not have the needed hardware in time to generate results for the large web track. We only participated in the Small Web Track (Text Only and Text and Link Analysis). However, our system was designed for large datasets and the quality of the TREC8 results are not representative of the system. More recently we have experimented with larger datasets and we have come to the conclusion that link analysis can significantly increase the quality of the ranking of search engines, a conclusion that is shared by many others in the literature [BP98, PBMW98, Kle98, CDR⁺98, CDDG⁺98, CDG⁺99]. We will report these new results in a future publication.

References

- [BH98] Krishna Bharat and Monika R. Henzinger. Improved Algorithms for Topic Distillation in Hyperlinked Environments. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 104–111, August 1998.
- [BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, April 1998.
- [CDDG⁺98] Soumen Chakrabarti, Byron E. Dom, Prabhakar Raghavan David Gibson, Ravi Kumar, Sridhar Rajagopalan, and Andrew Tomkins. Experiments in Topic Distillation. In *ACM SIGIR Workshop on Hypertext Information Retrieval on the Web*, Melbourne, Australia, 1998.
- [CDG⁺99] Soumen Chakrabarti, Byron E. Dom, David Gibson, Jon M. Kleinberg, S. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Hypersearching the Web. *Scientific American*, June 1999.

- [CDR⁺98] Soumen Chakrabarti, Byron E. Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon M. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, April 1998.
- [DGK⁺99] Brian D. Davison, Apostolos Gerasoulis, Konstantinos Kleisouris, Yingfang Lu, Hyunju Seo, Wei Wang, and Baohua Wu. DiscoWeb: Applying Link Analysis to Web Search. In *Poster proceedings of the Eighth International World Wide Web Conference*, pages 148–149, Toronto, Canada, May 1999.
- [HHMN99] Monika R. Henzinger, Allan Heydon, Michael Mitzenmacher, and Marc Najork. Measuring index quality using random walks on the web. In *Proceedings of the Eighth International World Wide Web Conference*, pages 213–225, Toronto, Canada, May 1999.
- [Kle98] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA-98)*, pages 668–677, San Francisco, CA, January 1998. Expanded version at <http://www.cs.cornell.edu/home/kleinber/>.
- [PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Unpublished draft (no longer available online)., 1998.

SMART in TREC 8

Chris Buckley*, Janet Walz*,

Abstract

This year was a light year for the Smart Information Retrieval Project at SabIR Research and Cornell. We officially participated in only the Ad-hoc Task and the Query Track. In the Ad-hoc Task, we made minor modifications to our document weighting schemes to emphasize high-precision searches on shorter queries. This proved only mildly successful; the top relevant document was retrieved higher, but the rest of the retrieval tended to be hurt. Our Query Track runs are described here, but the much more interesting analysis of these runs is described in the Query Track Overview.

Basic Indexing and Retrieval

In the Smart system, the vector-processing model of retrieval is used to transform both the available information requests as well as the stored documents into vectors of the form:

$$D_i = (w_{i1}, w_{i2}, \dots, w_{it})$$

where D_i represents a document (or query) text and w_{ik} is the weight of term T_k in document D_i . A weight of zero is used for terms that are absent from a particular document, and positive weights characterize terms actually assigned. The assumption is that t terms in all are available for the representation of the information.

The basic "tf*idf" weighting schemes used within SMART have been discussed many times. For TREC 8 we made a slight modification to Lnu-ltu weights we have used in the past 4 years in TREC 4-7. We noticed that the pivoted byte-length document normalization used by Singhal et al in TREC 7([5]) seems to favor high precision searches when used with short queries. It is a bit more biased towards shorter documents than our previous "u" scheme which uses number of unique terms in the document, thus a good short document containing all the query terms will be ranked highly. We hoped that this would enable our blind feedback query expansion to be based on more relevant documents and thus improved.

The same phrase strategy (and phrases) used in all previous TRECs (for example [2, 3, 4, 1]) are used for TREC 8. Any pair of adjacent non-stopwords is regarded as a potential phrase. The final list of phrases is composed of those pairs of words occurring in 25 or more documents of the initial TREC 1 document set. Phrases are weighted with the same scheme as single terms. Note that no human expertise in the subject matter is required for either the initial collection creation, or the actual query formulation.

When the text of document D_i is represented by a vector of the form $(d_{i1}, d_{i2}, \dots, d_{it})$ and query Q_j by the vector $(q_{j1}, q_{j2}, \dots, q_{jt})$, a similarity (S) computation between the two items can conveniently be obtained as the inner product between corresponding weighted term vectors as follows:

$$S(D_i, Q_j) = \sum_{k=1}^t (d_{ik} * q_{jk}) \quad (1)$$

Thus, the similarity between two texts (whether query or document) depends on the weights of coinciding terms in the two vectors.

*SabIR Research, Inc.

The Cornell TREC experiments use the SMART Information Retrieval System, Version 13.3, and most were run on a dedicated Intel 350 Mhz Pentium running Linux, with 128 Megabytes of memory and 54 Gigabytes of local disk.

SMART Version 13 is the latest in a long line of experimental information retrieval systems, dating back over 30 years, developed under the guidance of G. Salton. The new version is approximately 48,000 lines of C code and documentation.

SMART is highly flexible and very fast, thus providing an ideal platform for information retrieval experimentation. Documents for TREC 8 are indexed at a rate of about 2 Gigabytes an hour, on hardware costing under \$2,000 new. Retrieval speed is similarly fast, with basic simple searches taking much less than a second a query.

Ad-hoc Task

The basic approach we used for this year's TREC ad-hoc task is almost identical to our TREC 5 approach. We only used one algorithm (no experimental algorithm this year!), and ran it on 4 different topic lengths. Our TREC 5 paper [1] gives the details and rationale for the approach. The only important difference is that we used the *Lnb* weighted documents described above instead of the *Lnu* documents of TREC 5.

The basic algorithm is

1. Retrieve 1000 documents using the initial query (using *Lnb.ltu* weights).
2. Generate cooccurrence information about the query terms from the top 1000 documents.
3. Rerank the top 50 documents as in TREC 5, using correlation between query terms and other terms, as well as proximity information of the query terms.
4. Assume the top 20 documents relevant, documents ranked 501–1000 non-relevant.
5. Expand the query by 25 words and 5 phrases using Rocchio expansion with $\alpha = 8$, $\beta = 8$, and $\gamma = 8$.
6. Retrieve the final set of 1000 documents using the expanded query.

Ad-Hoc experiments and analysis

We submitted four runs in the ad-hoc category, all using the same algorithm. Sab8A1 used only the title field of the topic, Sab8A2 used only the description field, Sab8A3 used all three fields, and Sab8A4 used the title and description field (this last run being the requested “official” ad-hoc run).

Table 1 shows the results for the various runs across 50 queries. Results are all quite close to each other, with noticeable disagreements between the various evaluation measures.

Run	Average precision	Total rel retrieved	R precision	Precision @100 docs	Precision @Rcl(0)
Sab8A1(t)	.2553	3006	.2901	.2376	.7360
Sab8A2(d)	.2407	2829	.2829	.2228	.7988
Sab8A3(tdn)	.2546	2957	.3088	.2316	.8514
Sab8A4(td)	.2608	2986	.3021	.2384	.7860

Table 1: Ad-Hoc results (50 queries)

Table 2 shows that our runs are reasonably mediocre when compared with other runs using Average Precision. This is a disappointing performance; it is clear that the attempt to increase emphasize high

precision in the top retrieved documents by the use of Lnb document weights did not end up helping our expansion and therefore overall results.

Our approach does have a positive effect at the very high-end. Using Precision at Recall (0), which basically measures precision of the top retrieved document, our runs do considerably better than they do when evaluated using Average Precision. The .8514 figure for Sab8A3 in Table 1 is the second best figure among all automatic ad-hoc TREC runs, a mere 0.3% worse than the leading run. However, the Average Precision for Sab8A3 is a whopping 23.7% worse than the Average Precision for that same run. Given our high-end performance, there may be environments where it would be appropriate to use the algorithms here, but it is clear they should not be used for general retrieval.

Run	Task pool	Best	\geq median
Sab8A1	title	4	28
Sab8A2	title-desc	0	30
Sab8A3	entire	0	25
Sab8A4	title-desc	0	36

Table 2: Comparative automatic ad-hoc results (Av-Prec 50 queries)

Query Track

The Query Track is a bit different from other TREC tracks in that the individual participants all contribute data into a common pool for later analysis, rather than be evaluated separately. It is an attempt to examine the variability of queries by getting multiple query variations of past TREC topics and running them on different systems. The participants contribute their query variations, then run the variations from all other participants, and finally attempt to analyze the results.

The Query Track Overview in this proceedings gives the overall goals and procedures for the track, and gives all the analysis of the pooled results done so far. This section merely goes into the details of our contributions to the common pool of data and assumes the reader has background knowledge of the task itself.

Query Variations

We contributed 4 versions of each of the 50 topics of the TREC 1 Ad-hoc Task. All versions were created by the same person; an expert system designer of SMART. For each topic

1. Sab3a: The user looked at the results of a retrieval (done with the SMART TREC 4 algorithm, see below) using that topic as a query. In general, 3 or 4 relevant documents were looked at. Occasionally, some of the non-relevant retrieved documents were also examined. Just given these documents, a one sentence query was formed. The topic itself was not examined.
2. Sab1b: After constructing the sentence above, the user then constructed a 2-3 word short query, again before looking at the topic.
3. Sab1a: The user constructed a 2-3 word short query after looking at the topic. The user had memories of the relevant documents seen in the construction of the above two query sets.
4. Sab1c: An automatic blind feedback run was made on the original topics. The 2-3 most highly weighted terms/phrases in the expanded query were manually de-stemmed and formed into a query.

Including everything, it took 5 hours to construct the 4 query sets above. About 2 hours of that was setting up and running the indexing and retrievals on the Query Track learning and test sets.

Retrieval Variations

There were a total of 23 query sets (each composed of 50 queries) submitted by the participating groups. 21 of those were natural language and 2 were expanded lists of weighted terms.

We ran three different retrieval algorithms on each of the 23 query sets. Each algorithm used the basic SMART TREC 4 approach:

- Documents indexed with adjacency phrases and weighted with “Lnu” scheme.
- Queries indexed with adjacency phrases and weighted with “ltu” scheme (The two weighted term query sets used the given weights and did not have any phrases added.)
- If expansion terms needed, perform blind relevance feedback assuming the top 20 retrieved documents are relevant. Add and reweight terms.
- Run (possibly reweighted) queries against test documents, retrieving 1000 documents, and submit run to NIST.

The three algorithms were deliberately kept simple to aid in the later analysis, and so there would be no problems running all of the needed runs. None of the algorithms represent the best that we can do. The three run approaches are:

1. Saba: Index the query using terms from the query plus any adjacency phrases occurring in it.
2. Sabm: Moderate blind feedback expansion. Add 5 single terms and 2 adjacency phrases from the top documents.
3. Sabe: Blind feedback expansion. Add 50 single terms and 10 adjacency phrases from the top documents.

We turned in a total of 69 retrieval run results to NIST (23 query sets * 3 run approaches). In September we got the results from all the participants (about 450 Mbytes of results) and started our analysis! Analysis from all the groups is presented in the Query Track Overview.

Comparison with past TREC's

We performed our annual comparison of how TREC and our systems have varied over the years. We ran our 8 TREC SMART systems against each of the 8 TREC ad-hoc tasks.

Table 3 gives the results. Note that the indexing of the collections has changed slightly over the years so results may not be exactly what got reported in previous years. In the interest of speed, we ran our current implementation of the query and document indexing and weighting.

Comparing the columns of Table 3 gives an indication of how the difficulty of TREC task has changed over the 8 years of TREC. For example, eight different versions of the same system all do from 45% to 65% worse, in absolute numbers, on the TREC 7 task as compared to the TREC 1 task. The TREC 1 and TREC 2 figures are about the same. Performance starts to drop in TREC 3 and 4 when the queries get progressively shorter. The short high-level queries of the last 4 TRECs prove more difficult for all versions of SMART. All the methods agree that the TREC 8 task is a bit easier than any of the TREC 5-7 tasks.

Looking at other evaluation measures (not given here) run on the same 8x8 grid, we get confirmation that our TREC 8 approach does well at the high end of the retrieval ranking while not doing well overall. For the TRECs with short queries, TREC 5-8, the TREC 8 approach had the highest evaluation numbers for measures Precision(5), Precision(10), Precision at Recall(0), Precision at Recall(.10), beating all other 7 approaches. But Average Precision as given in Table 3) is the lowest in the past 4 years for all tasks except TREC 6. Other measures such as the total relevant retrieved agree with Average Precision

Methodology and Run	TREC 1 Task	TREC 2 Task	TREC 3 Task	TREC 4 Task	TREC 5 Short	TREC 6 DESC	TREC 7 DESC	TREC 8 TI-DES
TREC 1: ntc.ntc	.2442	.2615	.2099	.1533	.1048	.0997	.1137	.1412
TREC 2: Inc.ltc	.3056	.3344	.2828	.1762	.1111	.1125	.1258	.1846
TREC 3: Inc.ltc-Exp	.3400	.3512	.3219	.2124	.1287	.1242	.1679	.2102
TREC 4: Lnu.ltu-Exp	.3628	.3718	.3812	.2773	.1842	.1807	.2262	.2436
TREC 5: Exp-rerank	.3759	.3832	.3985	.3128	.2047	.1844	.2543	.2629
TREC 6: Rrk-clust	.3711	.3779	.4014	.3037	.2031	.1768	.2512	.2654
TREC 7: Rrk-clust	.3779	.3837	.4002	.3137	.2116	.1804	.2543	.2679
TREC 8: Lnb	.3563	.3623	.3647	.2836	.1997	.1857	.2282	.2608

Table 3: Comparisons of past SMART approaches with present

Cross-Language

We ended up submitting one unofficial run to the Cross-language Track. The NIST organizers asked us to submit an Italian query on Italian documents run in order to augment the Italian language pool for the full Cross-language task (there were extremely few Italian documents retrieved by any of the groups.) We discuss it here just to document where those relevant documents came from.

Words were stemmed with our multi-lingual stemmer, which includes a very few Italian specific rules:

- Removes initial "all", "d", "dall", "dell", "l", "nell", "quest", "sull", "un".
- Removes final "a", "e", "i", "o", "mente".

Documents and queries were indexed with single terms, but no phrases.

The retrieval algorithm used is exactly the same as the TREC4 algorithm used for the Query Track Sabe run described above, except no adjacency phrases were used. Again, a simple run was deemed the best.

NIST requested the run of us since we could supply it quickly. Indeed, it took us a bit under 2 hours, which included fetching the queries via ftp, setting up and indexing the documents, doing the retrieval and ftp-ing the results to NIST.

Conclusion

We participated in TREC 8, though at a slightly lower level than in previous years. Much of the interesting work we did for TREC 8 appears in the Query Track Overview.

Our ad-hoc runs this year turned out fairly mediocre. We tried to develop a high-precision approach in the hopes that blind feedback query expansion would then perform well, but we were unsuccessful. Our approach did result in good performance at the high end, but our overall performance was poor. We did not retrieve nearly as many total relevant documents as either other TREC 8 groups did, or as we would have retrieved if we had used one of our systems from the past few TRECs.

References

- [1] Chris Buckley, Mandar Mitra, Janet Walz, and Claire Cardie. Using clustering and superconcepts within SMART : TREC 6. In E. M. Voorhees and D. K. Harman, editors, *The Sixth Text REtrieval Conference (TREC-6)*. NIST Special Publication 500-240, 1998.
- [2] Chris Buckley, Gerard Salton, and James Allan. Automatic retrieval with locality information using SMART. In D. K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 59–72. NIST Special Publication 500-207, March 1993.

- [3] Chris Buckley, Amit Singhal, and Mandar Mitra. New retrieval approaches using SMART : TREC 4. In D. K. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*. NIST Special Publication 500-236, 1996.
- [4] Chris Buckley, Amit Singhal, and Mandar Mitra. Using query zoning and correlation within SMART : TREC 5. In D. K. Harman, editor, *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*. NIST Special Publication 500-238, 1997.
- [5] Amit Singhal, John Choi, Donald Hindle, David D. Lewis, and Fernando Pereira. AT&T at TREC-7. In E. M. Voorhees and D. K. Harman, editors, *The Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication 500-242, 1999.

SCAI TREC-8 Experiments

Dong-Ho Shin, Yu-Hwan Kim, Sun Kim,
Jae-Hong Eom, Hyung-Joo Shin, Byoung-Tak Zhang

Artificial Intelligence Lab (SCAI)

Dept. of Computer Engineering

Seoul National University

Seoul 151-742, Korea

E-mail: {dhshin, yhkim, skim, jheom, hjshin, btzhang}@scai.snu.ac.kr

ABSTRACT

This working note reports our experiences with TREC-8 on four tracks: Ad Hoc, Filtering, Web, and QA. The Ad Hoc retrieval engine, SCAIR, has been used for the Web and QA experiments, and the filtering experiments were based on its own engine. As a second entry to TREC, we focused this year on exploring possibilities of applying machine learning techniques to TREC tasks. The Ad Hoc track employed a cluster-based retrieval method where the scoring function used cluster information extracted from a collection of precompiled documents. Filtering was based on naive Bayes learning supported by an EM algorithm. In the Web track, we compared the performance of using link information to that of not using the information. In the QA track, some passage extraction techniques have been tested using the baseline SCAIR retrieval engine.

1 Introduction

In TREC-8, SCAI participated in 4 different tracks: Ad Hoc, Filtering, Web and QA. Among these only the Ad Hoc track is the second entry, others are the first participation. The Ad Hoc, Web, and QA tasks have been based on the SCAI information retrieval engine, SCAIR. Due to the different characteristics, the Filtering experiments were based on its own engine and storage system. After the release of the TREC-7 results, we experimented in various ways to get more experiences. This year our focus has been exploring various possibilities of using machine learning algorithms to improve the performance on TREC tasks.

Our retrieval engine, SCAIR, has been upgraded from the experiences of last year. SCAIR manages documents in inverted file structure and supports some convenient APIs to higher applications. A number of 556 stop words and 335879 indexing terms are used. Porter's stemming algorithm [4] and a modified suffix truncation algorithm are implemented. These are not for indexing but for retrieval, so more flexible retrieval is possible by setting some parameters. WordNet is also embedded to manipulate query terms.

This paper is organized as follows. In Section 2, we describe and report on the Ad Hoc exper-

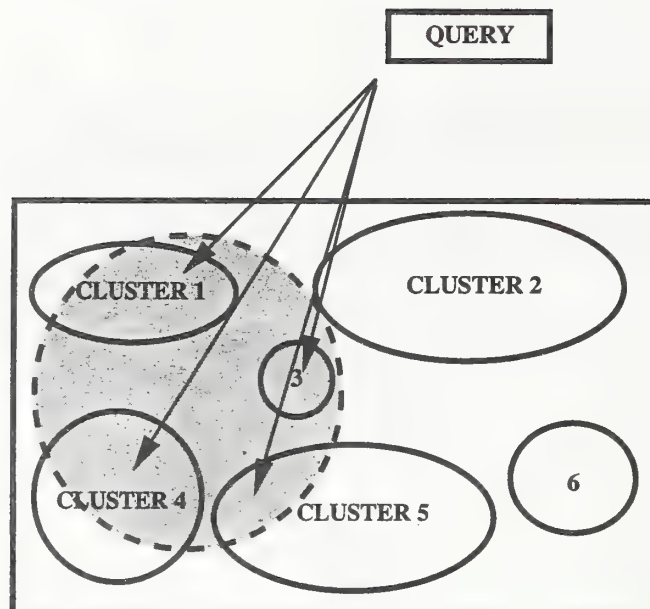


Figure 1: Document clusters and the Ad Hoc retrieval

iments. Section 3 discusses the result of the filtering track based on naive Bayes classifiers trained with expectation maximization (EM). Section 4 analyzes the effect of using link information in the Web track. Section 5 reports on the experimental results of the QA task. Section 6 summarizes our experiments at TREC-8.

2 Ad Hoc Track

Our Ad Hoc retrieval is based on the vector space model. Formally, a document is represented as a list of terms or vectors. A document collection is represented as a term-document matrix which are normally very sparse. A query consists of terms, too.

The documents are indexed by the classical $tf \cdot idf$ weighting scheme:

$$w_{ij} = tf_{ij} \cdot \log \left(\frac{N}{df_j} \right), \quad (1)$$

where w_{ij} is the weight of j th term in the i th document, tf_{ij} is the frequency of the j th term in the i th document, N is the total number of documents in the collection, and df_j is the number of documents in which the j th term occurs. Query terms are weighted only by the idf value.

The similarity between a document and a query is measured by cosine coefficient:

$$S_{ij} = sim(d_i, q_j) = \frac{\sum_{k=1}^n w_{ik} \cdot q_{jk}}{\sqrt{\sum_{k=1}^n w_{ik}^2 \cdot \sum_{k=1}^n q_{jk}^2}} \quad (2)$$

where w_{ik} and q_{jk} are term weights for document i and query j , respectively. After query-document similarities are measured, an ordered list of documents is produced. Clustering information was then used to rearrange the list.

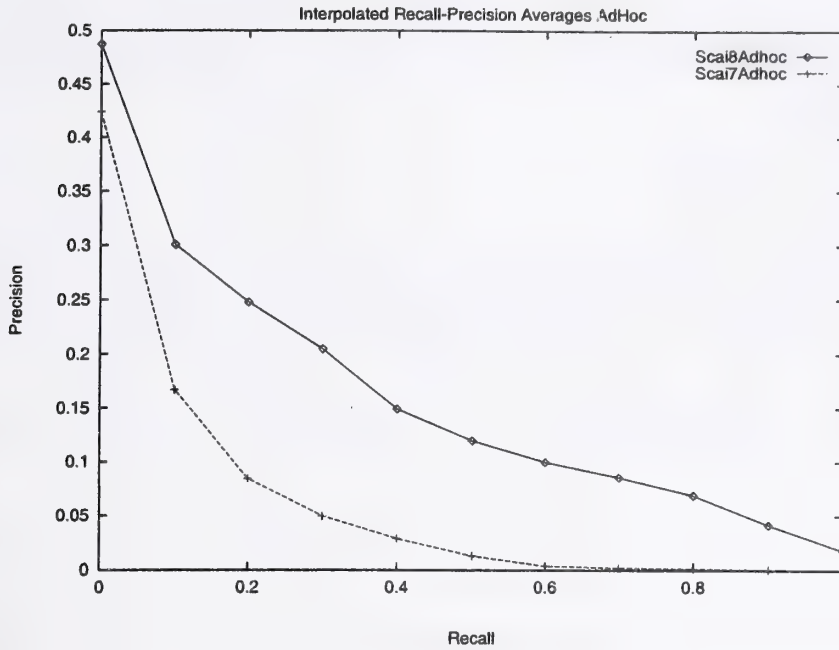


Figure 2: Ad Hoc results

The cluster information was obtained from a precompiled collection of documents. Figure 1 illustrates the relation of clustering and retrieval. The rationale behind this approach is as follows. In general retrieval situations, documents are retrieved from multiple clusters due to query ambiguity. But user's information needs are usually concentrated on one category. Thus, by excluding non-relevant clusters, retrieval performance might be improved.

A hierarchical clustering method is used to partition the whole document collection into a number of disjoint subcollections. Let y_i denote the score of document i . Then, the average score of the original collection of N documents are

$$\bar{Y} = \frac{\sum_{i=1}^N y_i}{N}. \quad (3)$$

After clustering, the average score of cluster k to which document i belongs is given as

$$\bar{c}_k^i = \frac{\sum_{i \in C_k} y_i}{N_k}, \quad (4)$$

where C_k is the k th cluster and N_k is its size. Using this information, the similarity of document d_i to query q_j is now updated by

$$S'_{ij} = S_{ij} \frac{\bar{c}_k^i}{\bar{Y}} \quad (5)$$

Our assumption is that the top N documents retrieved are relevant, and the clusters which have many relevant documents are considered as relevant clusters. This mechanism gives higher scores to the documents that belong to the relevant clusters, though they are not selected into the top N list. In the experiments we set $N = 100$. This mechanism is similar to 'winner and his neighbor

gang take all' rule. Figure 2 compares our TREC-7 vs. TREC-8 Ad Hoc results. As can be seen in the figure, a significant improvement in Ad Hoc performance was achieved by using the additional cluster information.

3 Filtering Track

We participated in the batch filtering track. As the first entry to the filtering track in TREC, our main concern this year was to explore the possibilities of machine learning algorithms for information filtering. We experimented with naive Bayes classifiers trained with the expectation-maximization algorithm [2].

Naive Bayes is a statistical approach to tackle classification problem. In the Bayesian approach, the most probable target value is assigned to the new document.

$$P(c_k|d_i) = \frac{P(c_k)P(d_i|c_k)}{P(d_i)} = \frac{P(c_k) \prod_{t=1}^{|d_i|} P(w_{it}|c_k)}{P(d_i)} \quad (6)$$

where d_i is the i th document, c_k is the k th class, and t is the index over the terms in d_i . Naive Bayesian classifiers assume that terms are conditionally independent of target value, thus assuming the second equality in the above equation. EM algorithm has been used to improve the performance of this algorithm by using unlabeled documents for getting more reliable statistics. Because TREC data has only small number of labeled documents and almost all the documents have no labels, EM algorithm may be useful in this situation.

We used no special techniques for preprocessing. Stop-words were removed and words were stemmed by the Porter's algorithm. The usual $tf \cdot idf$ was used to weight the terms. Cosine normalization was used. We used only FT92. No other documents, such as FT91, or thesaurus were used.

Figure 3 plots the LF1 values for our experiments. The result is moderate, averaging 4.16. Sometimes, naive Bayes with EM scored much higher than the average of the whole runs except the worst. In topic 352 and topic 389, for example, we achieved highest score, each 149 (followed by 130) and 218 (followed by 176). Usually, they had many positive examples. It can be concluded that naive Bayes with EM can achieve better performance with many positive examples.

4 Small Web Track

In our entry to the small Web track, our concern centered around the following two questions: 1. Do the best methods in the TREC Ad Hoc task also work best on the Web data (WT2g collection), and 2. Can link information in Web data be used to obtain more effective search rankings than can be obtained using page content alone. We first applied the method that was used for the Ad Hoc task to the Web task and then re-ranked its ranking results using link information.

We used a relatively simple re-ranking method. The top-ranking 2000 documents were chosen for re-ranking. We assumed that a document d and the set of documents, D_i , which are linked to d_i belong to the same class. We used only inlink documents (the documents that have links to

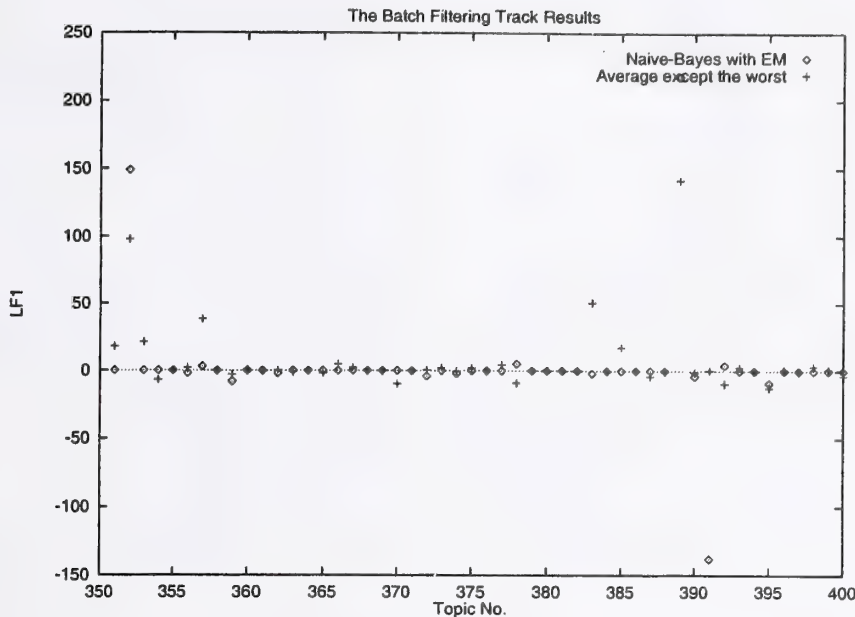


Figure 3: Batch filtering results.

document d_i), not outlink documents (the documents which document d_i has links to). This was motivated by the object-oriented concept where the objects in a higher class contains the objects in lower classes. We tested various re-ranking methods and made some preliminary experiments. In the following we report on a method that was used to get the results we submitted to TREC-8 Small Web track.

Let y_i be the score of document i . Let ℓ be the index of inlink documents of d_i , and y_ℓ its score. Then, the score of document d_i is updated by using the scores of inlink documents as follows:

$$y_i \leftarrow y_i + \alpha \left(\sqrt{\frac{L_i}{\sum_{\ell=1}^{L_i} y_\ell - y_i}} \right), \quad (7)$$

where α is a small constant.

Figure 4 shows the result of runs using the re-ranking method described above. The results for using contents only and for contents combined with link information are compared. The result of the Ad Hoc task is also shown. It can be observed that there is no significant difference in using link information for the Web documents. It seems that effective use of link information is not so trivial. An assumption in our use of link information was that the main topic of a document and its inlink documents are the same or very similar. It turns out, however, that the whole links are not always so closely related to each other and thus care must be taken to choose linked documents which are most significant. It can be said that the accuracy of the Ad Hoc engine influences the contribution of link information.

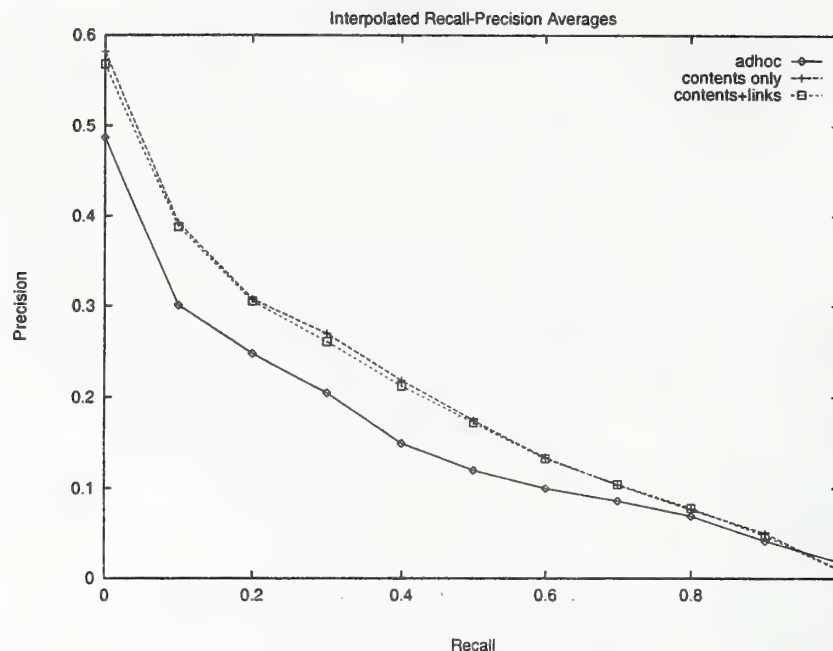


Figure 4: Small Web track results

5 Question Answering Track

The aim of the QA track is to get ‘information’ not ‘documents’ as retrieval results. Phrases are extracted for the user query. To achieve this purpose, we used a two-stage retrieval procedure. In the first stage, relevant documents for a user query are retrieved by the ad hoc retrieval engine. In the second stage, relevant passages are extracted from those documents by text-snippet extraction.

Text-snippets are extracted by determining local contexts, i.e. the phrases near the position of the highest-weighted query term. We refer to these contexts as answering zones. According to the QA rule, each candidate answering zone is 250 bytes long. More than one answering zones can be constructed within a document. If other important terms co-occur within the answering zone, the candidate zone get a high score. Then, the answering zone with the maximum score is selected as the final answer.

The experimental results were not very surprising, considering the difficulty of the task and our limited experience in this domain. We obtained 44 correct answers out of 198 questions. By correct we mean the answers were among the top-five candidate answers officially provided by TREC-8 QA track. About half of the correct answers were within top-two answers. Figure 5 analyzes the difference of our score from the median value for the 25 QA runs (entries) on each question. The difference of 1 for the questions in Figure 5 means that the QA runs from other entries found the correct answer whereas ours did not make it. The value of 0 indicates that our system achieved better results than other entries. Intermediate values indicate the relative degree of goodness of our results compared with other entries to QA task this year. It seems that the quality of QA accuracy seems very much dependent on the quality of the ad hoc retrieval engine.

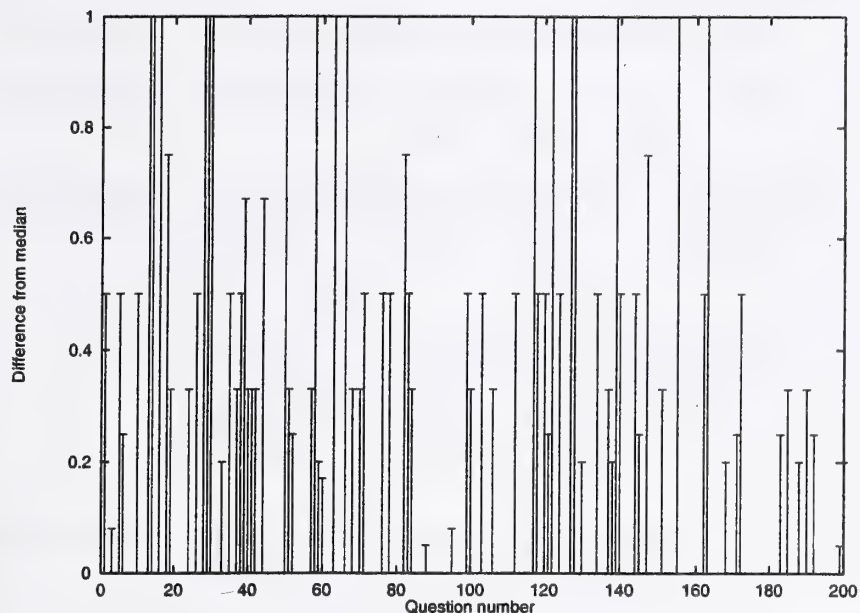


Figure 5: QA track results

6 Conclusions

In the Ad Hoc task, we achieved a significant performance improvement over our results obtained last year. It seems that the use of cluster information in the retrieved documents, in addition to the document-query similarity, is useful for enhancing precision of retrieval. In filtering, we tested the possibility of naive Bayes learning and achieved improved performance by using an EM algorithm, though the absolute performance was not very satisfactory. Our limited experiments in the small Web track showed the importance of a proper use of link information. The QA track performance was not satisfactory but our approach based on 'answering zones' seems promising.

Acknowledgements

This research was supported in part by the Korea Ministry of Information and Telecommunications under Grant C1-98-0068-00 through IITA.

References

- [1] Boyan, J., Freitag, D. and Joachims, T., A Machine Learning Architecture for Optimizing Web Search Engines, *AAAI-96 Workshop on Internet-based Information Systems*, 1996.
- [2] McCallum, A., Nigam, K., and Thrun, S., Learning to Classify Text from Labeled and Unlabeled Documents, *Machine Learning*, 1999.
- [3] Picard, J., Modeling and Combining Evidence Provided by Document Relationships Using Probabilistic Argumentation Systems, *SIGIR-98*, pp. 182-189, 1998.

- [4] Porter, M.F., An Algorithm for Suffix Stripping, *Program*, 14(3), pp. 130-137, 1980.
- [5] Robertson, S.E. and Sparck Jones, K., Relevance Weighting of Search Terms. *Journal of the American Society for Information Science* 27, 1976.
- [6] Rocchio, J., Relevance Feedback Information Retrieval, In G. Salton, editor, *The Smart Retrieval System - Experiments in Automatic Document Processing*, Prentice-Hall, pp. 313-323, 1971.
- [7] Silverstein, C. and Pedersen, J.O., Almost-Constant-Time Clustering of Arbitrary Corpus Subsets, *SIGIR-97*, pp. 60-66, 1997.
- [8] Schütze, H. and Silverstein, C., A Comparison of Projections for Efficient Document Clustering, *SIGIR-97*, pp. 74-81, 1997.
- [9] Yang, Y., Noise Reduction In A Statistical Approach To Text Categorization, *SIGIR-95*, 1995.

TREC-8 Experiments at SUNY at Buffalo

B. Han R. Nagarajan R. Srihari M. Srikanth

Department of Computer Science and Engineering
State University of New York at Buffalo,
Amherst, NY 14228-2567

1 Introduction

For TREC-8, State University of New York at Buffalo(UB) participated in the ad-hoc task and the spoken document retrieval(SDR) track. This is our first year of participation at TREC. We submitted two runs for the Ad-hoc task. The first run was term vector-based using SMART[10]. The second run used the TROVE - Text Retrieval using Object Vectors - system. For the SDR Track, we participated in the IR component of the Quasi-SDR task.

2 Ad-hoc Task

2.1 Overview

The UB team submitted two runs for the Ad-hoc Task. In the first run (UB99SW) we used SMART for indexing and retrieval on expanded queries generated by WordNet[4]. For each topic SMART+WordNet generated 5000 top ranked documents, which were input to our second retrieval engine TROVE (Text Retrieval using Object Vectors), and the result was submitted (UB99T). TROVE is our first attempt at exploring the feasibility of employing natural language processing (NLP) techniques in IR tasks. The system is implemented in its entirety from basic principles.

For decades NLP has been a promise to improving IR preformance, yet different experiments have had varying degrees of success [11]. In our experiment we focus on extracting semantics of documents using NLP techniques. In particular we avoid the ambiguities in full syntactical parsing and only extract semantics of partial phrases. The similarity between two phrases is then determined by the concepts they convey, instead of by their mere surface forms. The approach is similar to [12], however we analyze and represent the relations between phrase constituents by using semantics rules based on the types of constituents.

In the proposed TROVE system syntactical groups (noun groups, verb groups, etc) are first identified as *objects*, then short phrases representing semantics are grouped on top of these objects. The semantics are extracted based on the type codes assigned to the objects, and represented as *relation vectors* between two objects. The matching proceeds as a two-level process. In the first level match (*node-level* match) we establish a one-one mapping between objects in documents and objects in queries by comparing their similarities based on their semantic distance in WordNet. The second level match (*arc-level* match) proceeds by comparing the relation vectors between the corresponding objects. A similarity score is finally computed based on a conditional probability formula relating the two levels. The overall system diagram is given in Figure 1, where SEM denotes the semantics files, OBJ denotes the object list files, and VEC represents the vector files.

Due to time constraints, we were unable to finish the implementation of the complete TROVE system before the TREC deadline date. In particular the Semantics Interpreter and the Named Entity Tagger module were not ready. UB99T was generated using a partial implementation, i.e., only the node level match has been achieved, and the run was not completed.

For UB99T, we were able to use only node level matching for retrieval. Also, due to computational requirements, we were not able to complete the procedures for all 50 queries of the ad-hoc task. However,

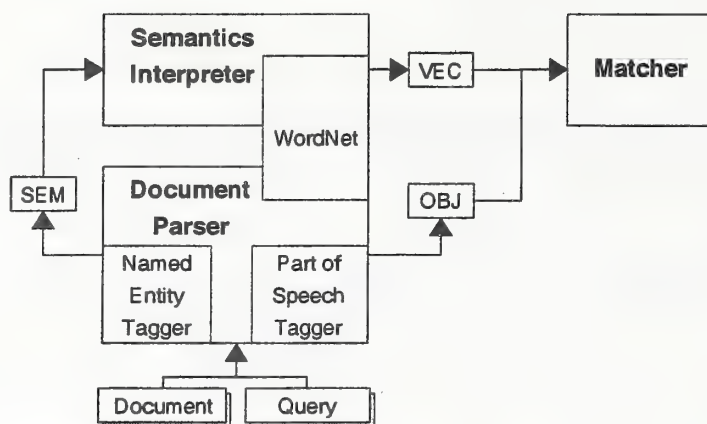


Figure 1: The TROVE system

in order to have our result evaluated, we decided to augment the incomplete portions of UB99T with the results from UB99SW. The two runs were submitted for comparison purposes.

2.2 Query Expansion via WordNet

In UB99SW we used WordNet for simple query expansion. For each topic we used the title and the description and filtered the stop-words out. Then for each noun and verb¹ we traversed upward and downward via hyponyms/hypernyms semantic links. The depth of traversal was arbitrarily set to 7 for ancestors and 6 for descendants. For all expansion words we encountered, only the first 5 of them were selected. A sample query expansion for the words **eliminate** and **border** is shown below:

eliminate destroy kill discharge expel eject get_rid_of do_away_with obviate rid_of annihilate
border boundary bound bounds boundary edge boundary_line borderline

2.3 Object Identification

To identify objects within a document, we first used a rule-based Part of Speech Tagger [2] to tag each term in the tokenized document. The identification is done using regular expressions involving the POS tags, and consists of 6 possible patterns:

1. Don't care (DC) patterns: These include existential 'there', list item marker, modal words (e.g. 'can'), pronouns (including wh-pronouns) and wh-adverbs. DCs only serve as placeholders in order to correctly extract phrases at the later stages.
2. Auxiliary/stative verb group (A/SVG) patterns: These include various *progressive form* and *perfect tense* verb groups. For examples, "is badly injured", "is charged with", "has amassed", and "have been considered seriously". Whenever possible, we combine the ending preposition word with the head verb, e.g., "is taken off" is grouped as "is taken-off". This is done by looking up potential groupings in WordNet.
3. Simple verb group (VG) patterns: These cover the usual verb groups like "accurately target" and "look at". Similar to A/SVG the potential grouping with the ending preposition word is checked.
4. Noun group (NG) patterns: These cover complete noun groups like "the three leftmost blue boxes" and "finishing touches". Complex noun groups are identified by enumerating all possible combinations and looking them up in WordNet. For example, "New York" will be grouped as "New_York" and "hot dog" as "hot_dog".

¹In SDR track a similar approach is used for query expansion, however we included adjective expansion as well in SDR track.

5. Function word (FW) patterns: These include all preposition words, conjunctions ('or' and 'and') and the word 'to'. FW is crucial at the stage of phrase extraction and semantics interpretation.
6. Punctuation mark (PM) patterns: These cover all punctuation marks, including possessive endings ('s) and symbols (e.g., parentheses). PMs only serve as placeholders in order to correctly extract phrases at the later stages.

Each object consists of a head and a list of modifiers. A type code is assigned to each word in an object according to 25 unique noun beginners [6] and 15 unique verb beginners [3], the type of an object is determined by the type code of its head.

In designing the patterns we try to avoid conflicts between different pattern matches: In case of conflicts the pattern with a smaller pattern number is favored. For example, "*is holding hands*" is identified as an A/SVG instead of a NG.

2.4 Phrase Extraction

Based on regular expressions involving the list of objects identified for each document, short phrases are extracted. There are 5 phrase patterns:

1. Complex nominals (CN): These group two NGs into one complex nominal, e.g. "*Chicago hot_dog*".
2. Possessive forms (PF): These cover the short phrases such as "*individual's personality*".
3. Complete phrases (CP): These include a complete short phrases like "*a man run across the street*".
4. NGs with situations (NS): These cover incomplete phrases such as "*man from Mars*". NGs are meant to capture parts of CPs when CP extraction is not possible.
5. Conjunctive groups (CG): These include noun and verb conjunctive groups. Examples: "*he moves into and holds the 2nd place in the competition*" and "*Mary and Bill's wedding*".

The patterns are applied in the order shown above, in particular CN and PF patterns are rewriting patterns in order to extract phrases like "*comments from stock exchange's listings division*". Each phrase is represented by 5 constituents: *subject*, *action*, *object*, *subject situations* and *situations*, of which each of the first three is an object, and the last two are two lists of FWs and objects. Thus "*the deal with BMW does not adversely affect Honda's policies in Europe*" is represented as:

- Subject: *the deal*
- Subject situations: *with BMW*
- Action: *does not adversely affect*
- Object: *Honda's policies*
- Situations: *in Europe*

The type of a particular constituent is determined by the type of the object involved. Therefore the semantics of a phrase can be inferred from the type information of its constituents.

2.5 Node-level Similarity

To obtain the node-level similarity score of a document, we compute the semantic distances of noun and verb objects between documents and in queries using WordNet. Since an object consists of a head and a list of modifiers, the similarity between two objects is taken as a linear combination of the similarity of the heads and that of the modifiers. Thus the problem is reduced to computing the similarity between two words.

The similarity of two words is determined by a slightly modified formula from [1]:

$$Sim(w_1, w_2) = \frac{idf_{w_1} \times idf_{w_2}}{Dist(w_1, w_2) + 1}$$

where $Dist(w_1, w_2)$ is the distance between word w_1 and w_2 , and idf_{w_i} is the *inverse document frequency* of word w_i in the data collection. The distance between two words is determined by a weighted edge count in WordNet following hypernym/hyponym links. The weighting scheme adopted reflects the likelihood of a particular sense of a word, together with the specificity of the words along the semantic paths based on the notion of *basic-level lexicalized concepts* [9].

For words of different syntactic categories (i.e., noun vs. verb), a conversion to noun is attempted for the verb by trying to find if it has a noun entry in WordNet. The similarity is then computed between the two nouns, but it is panelized by a predefined factor.

The final node-level similarity score for a document is thus defined as

$$Sim = Sim_R \times C_Q \times c \times C_D$$

where Sim_R is the accumulated words similarity (*raw* similarity), C_Q and C_D are the percentage of terms being matched in the query and the document (*coverage*), respectively, and c is simply a constant for weighting C_D .

2.6 Discussions

Since UB99T run was incomplete and it was augmented with results from UB99SW, it is rather difficult for us to draw any significant conclusion at the moment. Moreover, the augmentation implies a performance upper bound set by UB99SW. However, the preliminary results from TREC-8 evaluation shows that out of 28 topics processed by TROVE, 12 have been improved over UB99SW in terms of average precision, on average however, the performance degenerated. A closer analysis reveals that the lack of query term weighting misled the system to target the wrong content words in the data collection. This will be addressed in future work.

3 SDR Task

3.1 System description

An in-house version of term vector based retrieval system was used for TREC SDR experiments. A combination of query term expansion and blind relevance feedback[10, 5] was used to obtain our submissions – **cedar-r1** and **cedar-b1**.

The following similarity measure was used for matching queries and documents.

$$sim(q, d) = \sum_{t \in q \cup d} w(d, t) * w(q, t) \quad (1)$$

with

$$\begin{aligned} w(d, t) &= \frac{1}{Z(d)} f(d, t) \cdot \log\left(\frac{N}{f(t)}\right), \quad f(t) \neq 0 \\ w(q, t) &= \frac{1}{Z(q)} f(q, t) \end{aligned}$$

where $f(d, t)$ and $f(q, t)$ are the frequencies of the term t in document d and query q , respectively. The value of N gives the total number of documents in the collection and $f(t)$ gives the number of documents in which the term t occurs. $Z(d)$ and $Z(q)$ are normalizers for $w(d, t)$ and $w(q, t)$, respectively to ensure that the weights are between 0 and 1.

The runs **cedar-r1** and **cedar-b1** that correspond to the R1 and B1 retrieval conditions of SDR were obtained using the following approach. After some preprocessing of the documents, index terms are extracted

by filtering out stop words and reducing to word stems using the Porter stemming algorithm[7]. Around 540 stop words were used. The document retrieval is done in two phases. In the first phase, the query terms, which are stopped and stemmed, are used to rank the documents using the similarity measure in equation (1). In the second phase, the query is expanded using WordNet based on the query expansion scheme identified in section 2. The query expansion phase generated up to 5 terms for each query term in the original query. The original and expanded query terms are filtered and reweighted using the blind relevance feedback technique. The top 10 documents from the document ranking of the first phase are assumed to be relevant and the query term weight are adjusted accordingly. The reweighted set of query terms is used to rerank the documents in the collection. A final set of 1000 documents is retrieved using this document ranking.

3.2 SDR runs and Analysis

Table 1 gives the results for TREC-8 SDR submissions. It gives the precision values at different recall points as well as the average precision.

runid	5 docs	10 docs	20 docs	200 docs	Avg. Prec.
cedar-r1	0.4816	0.4551	0.3612	0.1247	0.3906
cedar-b1	0.4245	0.4000	0.3286	0.1127	0.3430

Table 1: TREC-8 SDR results

Table 2 gives a comparison of the performance of our system with respect to other participants. The table gives the number of queries that achieved the highest average precision, at least a median average precision or the lowest average precision.

runid	=Best	\geq Median
cedar-r1	2	13
cedar-b1	0	7

Table 2: Average Precision comparisons with other TREC-8 participants

Some of the reasons for this average performance of the system are:

- The query expansion used WordNet and up to 5 additional terms were added to the initial query for each query term. This strategy for query expansion seems insufficient from the performance. More terms would have improved the performance of the system.
- We used blind relevance feedback to reweight the WordNet expanded query terms. Reweighting the query terms did change the ranking of the results. An alternate strategy is to use some of the terms from the top 10 relevant documents retrieved based on the initial query as additional query terms.

We plan to experiment in these directions as well as use the TROVE system for SDR task.

4 Conclusions

This was our first participation in TREC. Our performance is just about the average performance of systems. Our participation in TREC was a learning experience. Based on the results, query expansion is one of the main areas we plan to concentrate on to improve our results.

With respect to TROVE, only a part of the system is currently implemented. The following is in the works for the future:

1. Semantics Interpretation based on short phrases with type information: Ideally this should provide an elegant rule syntax so that one can specify semantics rules for converting a short phrase into an appropriate 3-valued predicate, e.g.,

[location]/1 [location]/2 \Rightarrow IN(%2, %1) // example: *new_york suburb*
 [time]/1 [event]/2 \Rightarrow IN(%2, %1) // example: *yesterday's accident*
 [*]/1 [*]/2 [*]/3 \Rightarrow AGENT(%2, %1), PATIENT(%2, %3)

2. More sophisticated query processing: This includes a more detailed syntactical analysis for queries in order to filter out the unrelated words. Also the query term weighting/expansion can be done based on *blind relevance feedback* so a set of most similar terms will be returned as expansion candidates and the weight of a term which receives the highest hit score earns the highest weight.
3. Sense disambiguation via *class-based probability*: In current implementation sense disambiguation is done by weighting semantic paths in WordNet for a word such that the more frequently used senses are preferred. We plan to incorporate class-based probabilities [8] to compute the confidence score for each sense of a noun based on the pivotal verb. For example, the word 'snow' in "*snow is falling fast*" will have a much more preferred sense "*precipitation falling from clouds in the form of ice crystals*" instead of the sense for "*cocaine*", as hinted by the pivotal verb 'falling'.

References

- [1] Y. A. Aslandogan, C. Thier, C. T. Yu, J. Zou, and N. Rishe. Using semantic contents and wordnet(tm) in image retrieval. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, 1997.
- [2] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing, ACL*, 1992.
- [3] C. Fellbaum. A semantic network of english verbs. In C. Fellbaum, editor, *WordNet: an Electronic Lexical Database*, chapter 3. MIT Press, 1998.
- [4] C. Fellbaum, editor. *WordNet: an Electronic Lexical Database*. MIT Press, 1998.
- [5] W. B. Frakes and R. Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall Inc., Englewood Cliffs, NJ, USA, 1992.
- [6] G. A. Miller. Nouns in wordnet. In C. Fellbaum, editor, *WordNet: an Electronic Lexical Database*, chapter 1. MIT Press, 1998.
- [7] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.
- [8] P. Resnik. Wordnet and class-based probabilities. In C. Fellbaum, editor, *WordNet: an Electronic Lexical Database*, chapter 10. MIT Press, 1998.
- [9] E. Rosch, C. B. Mervis, W. Gray, D. Johnson, and P. Boyes-Braem. Basic objects in natural categories. *Cognitive Psychology*, 8:382 – 349, 1976.
- [10] G. Salton, editor. *The Smart retrieval system: experiments in automatic document processing*. Prentice-Hall, 1971.
- [11] A. F. Smeaton. Using nlp or nlp resources for information retrieval tasks. In T. Strzalkowski, editor, *Natural Language Information Retrieval*. Kluwer Academic Publishers, 1999.
- [12] T. Strzalkowski and et al. Natural language information retrieval: Trec-7 report. In D. K. Harman, editor, *NIST Special Publication 500-242: The Seventh Text REtrieval Conference (TREC-7)*, 1998.

CINDOR Conceptual Interlingua Document Retrieval: TREC-8 Evaluation.

Miguel Ruiz, Anne Diekema, Páraic Sheridan

MNIS-TextWise Labs
Dey Centennial Plaza
401 South Salina Street
Syracuse, NY 13202

Abstract:

The TREC-8 evaluation of the CINDOR system was based on English and French data from the cross-language retrieval track. Our objective was to continue our investigation of our conceptual interlingua approach to cross-language retrieval, specifically by measuring the contribution of conceptual retrieval over and above a baseline cross-language retrieval approach based on machine translation of queries. In both of the cross-language runs that were submitted for evaluation, corresponding to English-French and French-English retrieval, performance was measured at 75% of the equivalent monolingual searches. We noted however that absolute average precision values achieved were somewhat lower than many other systems in the cross-language track. Our hypothesis, that the underlying retrieval engine used in CINDOR was employing a simple retrieval function that was impacting performance, was confirmed through experiments with the SMART system configured with several different retrieval settings. Taken together, our TREC-8 experiments point to the value of our conceptual interlingua approach to retrieval, but indicate that our retrieval algorithm must be brought up to date so that valid comparisons may be made to other approaches used in other cross-language systems.

1. Introduction

The CINDOR project at MNIS-TextWise Labs is pursuing a 'conceptual interlingua' approach to cross-language information retrieval, based on a conceptual lexical resource modeled around WordNet [Miller 1990]. WordNet synonym groups, 'synsets', are taken to represent concepts which we assume are essentially language neutral. We have constructed the conceptual interlingua resource around the WordNet hierarchy by linking equivalent synonymous terms in several languages into the synsets representing a given concept. To date we have extended the conceptual interlingua to French, Spanish and Japanese, achieving approximately 20% coverage of WordNet synsets in each language. Our goal however is not to achieve complete coverage of WordNet content in each language. One of the objectives of our evaluation efforts has therefore been to investigate the extent to which conceptual interlingua coverage translates to vocabulary coverage in a typical document collection. For example, 55% of term occurrences in the TREC French collection match into our conceptual interlingua, which has 18% synset coverage in French.

Much of our research in the CINDOR project has been directed toward understanding the performance of a conceptual resource such as ours for cross-language retrieval. In particular, our participation in the TREC-8 evaluation was directed at verifying performance improvements in the current version of the CINDOR system over that which was used in our TREC-7 experiments, given significant re-development of the system over the period in between. The CINDOR

system evaluated here sits on top of an Oracle database and interacts with the ConText text management system available with the Oracle relational database management system. While Oracle provides the data management capabilities, ConText supports text indexing and retrieval over Oracle data. On top of this, CINDOR provides text processing to extract indexing terms and map them into our conceptual interlingua vocabulary for indexing. An overview of this architecture is provided in Section 2.

For TREC-8 we submitted two official runs, restricting our attention to English and French – that subset of the cross-language track data which overlaps with our current research focus. Independently of TREC, we are conducting benchmark evaluation experiments of CINDOR in Spanish, using previous TREC test collections, and Japanese, using the new NACSIS test collection [Kando 1999]. A new feature of the CINDOR system used in TREC-8 experiments is the inclusion of the Systran machine translation system to provide automated translations of input queries. The machine translation output serves as an additional source of evidence for target-language query terms and is used to complement the conceptual translation provided through the conceptual interlingua. As part of our TREC-8 experiments, we have therefore had the opportunity to investigate and identify the usefulness of conceptual interlingua translations over and above those provided by Systran. A report and analysis of our TREC-8 experiments and performance across French and English is presented in Section 3.

The initial review of our results, together with some investigative experiments using TREC-7 cross-language track data, suggests that CINDOR retrieval performance is being negatively impacted by reliance on the standard Oracle ConText '*tf*idf*' retrieval weighting algorithm. This is consistent with the well-established observation over past TREC evaluation experiments in general; that the weighting scheme is a crucial component in overall system performance. We therefore establish, in Section 4, the extent to which our runs using CINDOR with ConText may be impacted by this retrieval problem in order to suggest the extent to which performance can be further improved.

2. The CINDOR System.

The CINDOR (Conceptual Interlingua Document Retrieval) system is cross-language text retrieval system capable of accepting a user's query stated in their native language and then seamlessly searching, retrieving, relevance ranking and displaying documents written in a variety of foreign languages. CINDOR allows users of the system to state queries in any of the supported languages (currently English, French, Spanish, and Japanese) and search and retrieve documents from any of the supported languages.

The CINDOR system adopts a unique approach to cross-language information management based on a language-independent conceptual representation known as a 'Conceptual Interlingua'. This facilitates direct mapping between the interlingual representations of documents and user queries in multiple languages, a substantial advantage over systems, which rely on pairwise translations between languages. The CINDOR approach also ensures that documents and queries are matched at the underlying concept level, rather than relying on exact word matches. Queries are specified as natural language expressions rather than as keyword lists, as are commonly used for example in Internet search engines. This conceptual matching of natural language queries is designed to enhance retrieval effectiveness over keyword-based systems, which rely on exact, matching of words or word stems.

Conceptual Interlingua

We use the term conceptual interlingua to refer to a knowledge base of language-independent concept representations. Our current conceptual interlingua is a hierarchically organized concept lexicon in which concepts are related through various lexical relations. Concepts in the hierarchy are considered to be essentially language neutral and are then linked to their relevant terminological instantiations in various languages, currently English, French, Spanish and Japanese. Our Conceptual Interlingua therefore consists of two separate resources, which we refer to as the conceptual resource and the (multilingual) terminological resources.

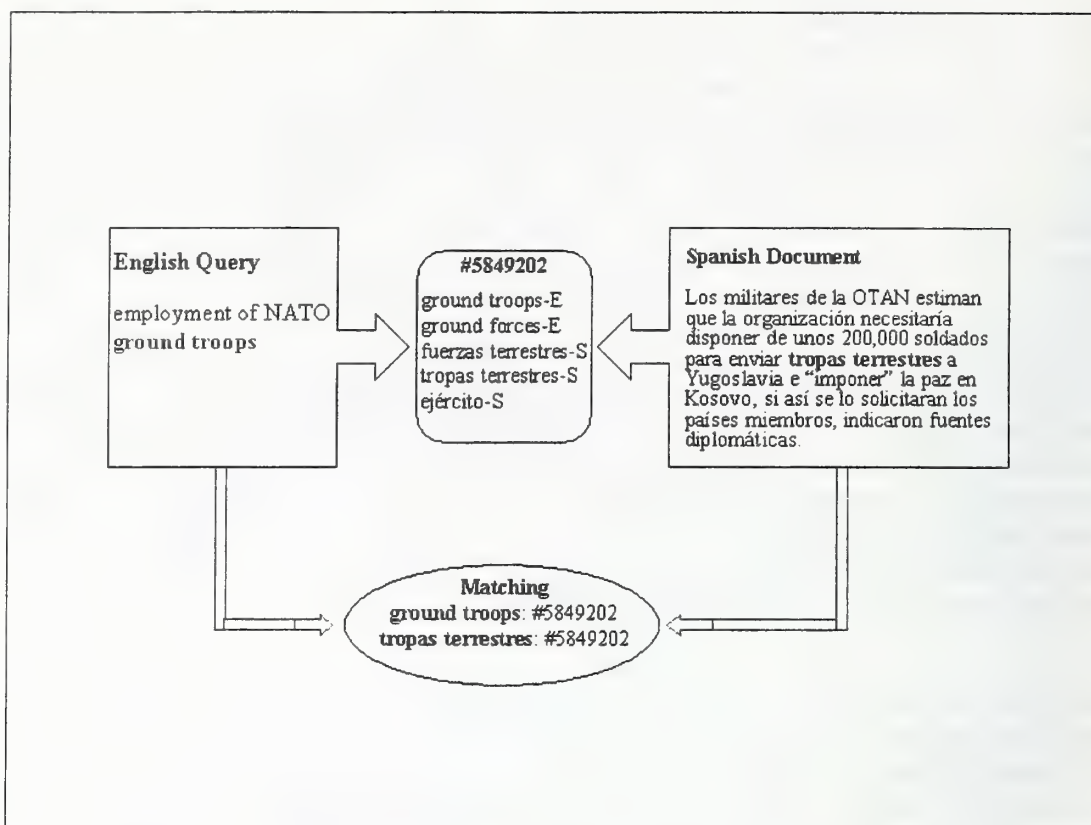
The CINDOR conceptual interlingua is built around WordNet [Miller 1990], a lexical resource, which contains approximately 165,000 different wordforms, organized into some 70,100 different concepts denoted by a group of synonyms, or 'synsets'. Starting from the English Princeton WordNet, a large portion of the synsets has been translated into French, Spanish and Japanese. The conceptual interlingua consists of synset numbers; i.e. for document indexing words are "translated" by their synset IDs. Our Interlingua is therefore set up so that equivalent words in English, French, Spanish and Japanese are indexed by identical synset IDs. We "cross the language barrier" by mapping everything to synset IDs.

Following the distinction between 'conceptual' and 'terminological' resources outlined above, the 'conceptual' resource of our conceptual interlingua consists of the WordNet hierarchy of synset labels. Each synset label (concept) is linked then to a set of words or phrases which instantiate that concept in each of the languages supported – the 'terminological' resource. For example, the concept of "*elasticity*: the tendency of a body to return to its original shape after it has been stretched or compressed", which has the label 131186, is instantiated in English and French as follows:

131186 spring, give, springiness
131186 élasticité, flexibilité, moëlleux

We consider the label 131186 to represent the language independent concept of *elasticity* so this number is part of our conceptual hierarchy. The terminology related to this concept in each language is then linked to this concept label from each of our language resources. In terms of CINDOR document processing, this means that any document or query term, which is identified as an instantiation of the concept of 'elasticity', is indexed to the concept label 131186. Whether the term occurs in an English, French, Spanish or Japanese document or query, the label will be the same and retrieval will be enabled, as illustrated in the Figure below using the term, "*ground troops*":

The architecture of the CINDOR system in its current form involves the use of the Oracle relational database management system (v8.0.5) with the ConText option for storage of source documents and management of inverted index tables. The conceptual interlingua resource is also transformed to a flat table and stored in an Oracle database. The Oracle system was chosen for its scalability and robustness, allowing CINDOR to be deployed over very large document collections and allowing for the full range of database management functions to be applied over stored text.



Example Cross-Language matching through Conceptual Interlingua

Integration of CINDOR functionality for cross-language retrieval is achieved in places where ConText provides for various "filters" to be applied to document content before being indexed and stored in inverted access tables. Although ConText filters were originally intended for simple format transformation, CINDOR subjects document content to full morphological analysis, part-of-speech tagging, and conceptual analysis against the conceptual interlingua. Word stems plus conceptual codes are returned to ConText from the CINDOR language analysis module and are then indexed into Oracle tables.

At retrieval time, natural language queries are analyzed in the same way as documents and are then transformed into SQL statements and submitted to ConText for evaluation. Through experimentation over time, we have found that the optimum query format consists of a number of query segments, each evaluated in turn and then combined into a final ranked result. To the extent that term and collection frequencies are computed and stored internally by ConText, the retrieval ranking algorithm is outside the control of the CINDOR system. We have augmented ConText retrieval through a standard document length normalization adjustment, but we have suspected for some time that internally ConText was using a rather simple weighting mechanism that could be substantially improved upon. This is a topic to which we have given some attention in our TREC-8 experiments.

3. TREC-8 Experiments.

TREC-7 represented the first evaluation of the CINDOR system, development of which was completed with little time to spare before submission of official runs [Diekema *et al* 1999]. The intervening year has seen a substantial re-development effort of many components of the CINDOR system, particularly with a view to addressing shortcomings identified in TREC-7 experiments. An important component of our objectives in TREC-8 therefore involves establishing the extent to which this re-development effort has lead to improvements in system performance.

A new component of CINDOR processing which has been introduced over the past year is a machine translation system, which is used both for translating queries into the language of documents, and also translating foreign language documents back to the language of the user on demand. The use of a machine translation system at query time is intended to contribute a further source of target language terms for queries and complement the conceptual mapping provided through the conceptual interlingua, especially in cases where query terms are not present in the interlingua resource.

Although the cross-language track again set as the main task the retrieval of documents from a multi-lingual set of English, French, German and Italian documents, we focused on the sub-task which involved the language pair of English and French since these are the two of the four which are covered by our resources. Two official runs were submitted to NIST for evaluation; English queries against French documents (TW8E2F) and French queries against English documents (TW8F2E). These official runs are complemented here by a series of other unofficial runs which were undertaken to allow us to examine a range of evaluation questions which were of interest.

A primary question of interest in evaluating the CINDOR system relates to the contribution of our conceptual interlingua approach to retrieval. Although designed primarily to facilitate cross-language retrieval, we anticipate that the benefits of synonym expansion may be observed also in monolingual retrieval settings. We have therefore completed experiments in which the use of conceptual interlingua indexing was de-activated for retrieval and compared performance to that of the standard CINDOR system with the conceptual interlingua enabled. A comparison between results for monolingual French retrieval is included in Figure 1, while English-French cross-language retrieval is illustrated in Figure 2.

The difference between the baseline system and the CINDOR system in Figures 1 and 2 is that the baseline system completes morphological and part-of-speech analysis but does no further processing, while CINDOR takes the further step of assigning conceptual codes to index terms. Further, these experiments included the current CINDOR proper name recognition module, which is still under development. This module attempts to recognize and tag proper names such as people, places, organizations etc. and to categorize them into appropriate classes. The advantage of this module in these experiments is likely to come from the ability to recognize multi-word proper names and to treat them as a single unit. The baseline system for cross-language retrieval consists of the baseline monolingual system augmented by Systran machine translation of queries for matching English queries against French documents.

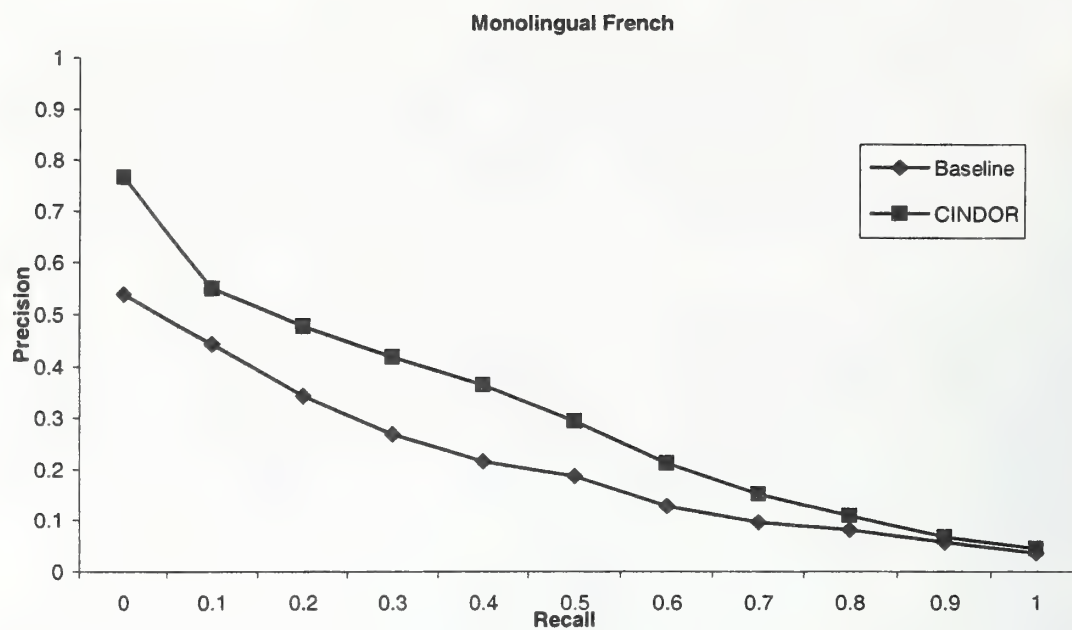


Figure 1: Conceptual Interlingua retrieval; French-French

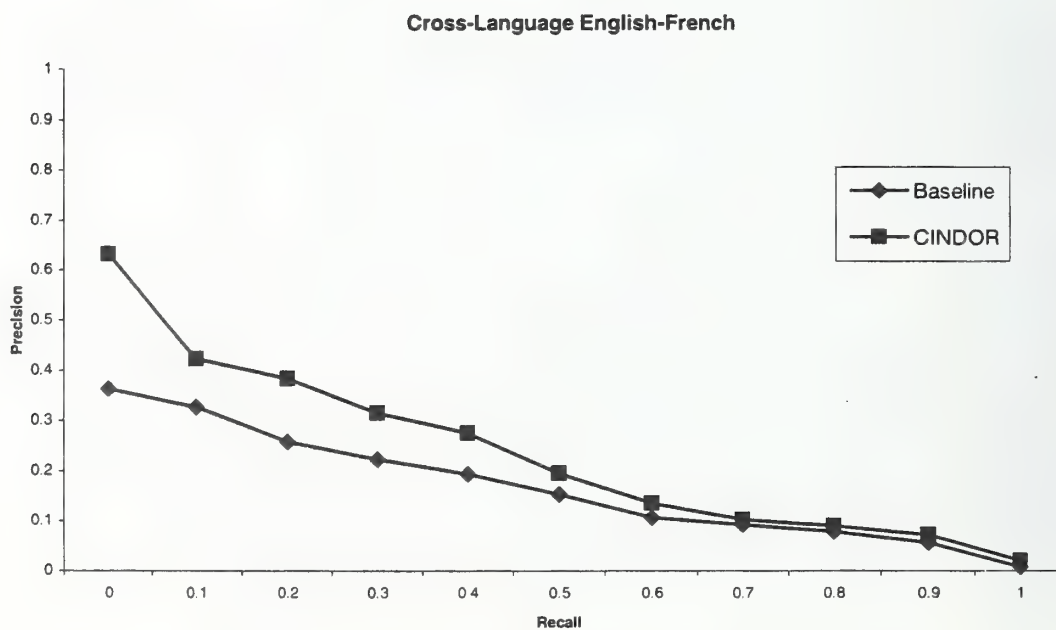


Figure 2: Conceptual Interlingua retrieval; English-French

In monolingual French retrieval, the baseline average precision is 0.2016, while CINDOR processing raises the value to 0.2921, a 45% increase. Similarly in our cross-language experiments, the baseline average precision is 0.1590, while CINDOR processing raises this to 0.2209, a 39% increase. This later result is particularly interesting because it represents the increase in cross-language retrieval performance to be achieved through the use of the conceptual interlingua above that achieved through straightforward translation of queries through automated machine translation. It is unclear from these experiments however, the extent to which performance gains above the baseline system may be attributed primarily to the conceptual matching enabled through the conceptual interlingua, or to the increased precision achieved through matching as single units the proper names which are frequent in TREC queries. In the ongoing evaluation of the CINDOR system, we will undertake in the near future a detailed analysis of the contribution of our proper name recognition module, which will enable us to determine exactly the contribution of each component.

One of the advantages of our conceptual interlingua approach to cross-language retrieval, at least in theory, is that by matching at the conceptual level, we can expect minimal loss in retrieval precision when matching across different languages compared to retrieval in a monolingual environment. The extent to which cross-language results mirror those of equivalent monolingual searches is easily facilitated using the TREC data, since topics are made available in each of the document languages. Our official submissions were intended to compare the performance of English-French retrieval compared to French-French monolingual, and French-English compared to monolingual English-English retrieval. The results of these runs are presented in Figures 3 and 4 below.

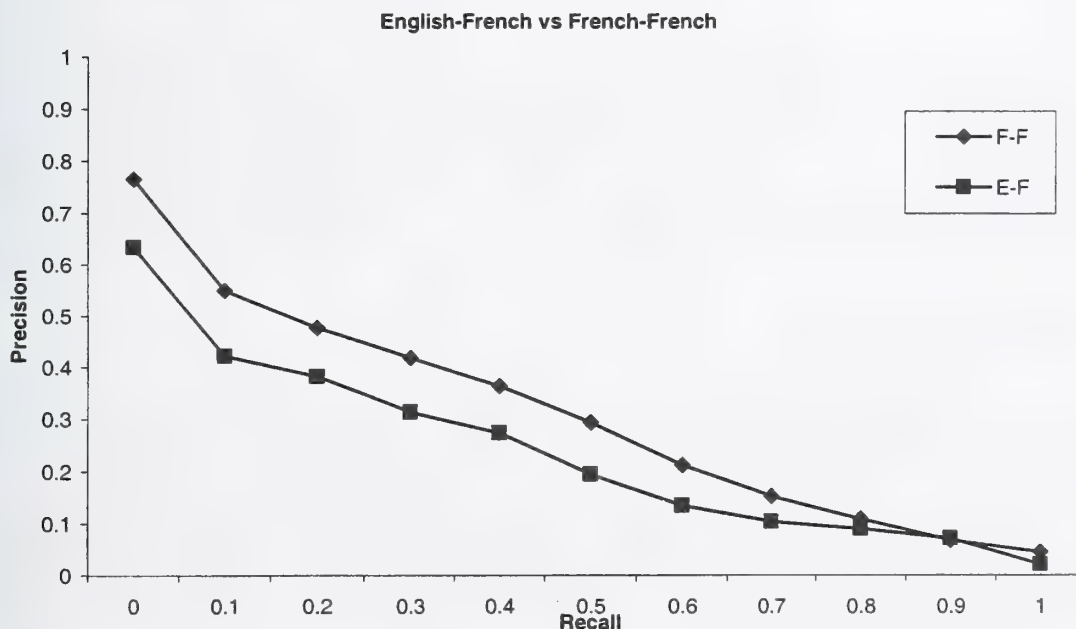


Figure 3: Cross-language versus Monolingual: French documents.

Comparing cross-language retrieval of French documents in response to English queries against the monolingual case where equivalent French queries are used, illustrated in Figure 3, indicates that our cross-language retrieval performance is at 75% of monolingual. The English-French run

has an average precision of 0.2209, compared to an average precision of 0.2921 for the monolingual French-French run.

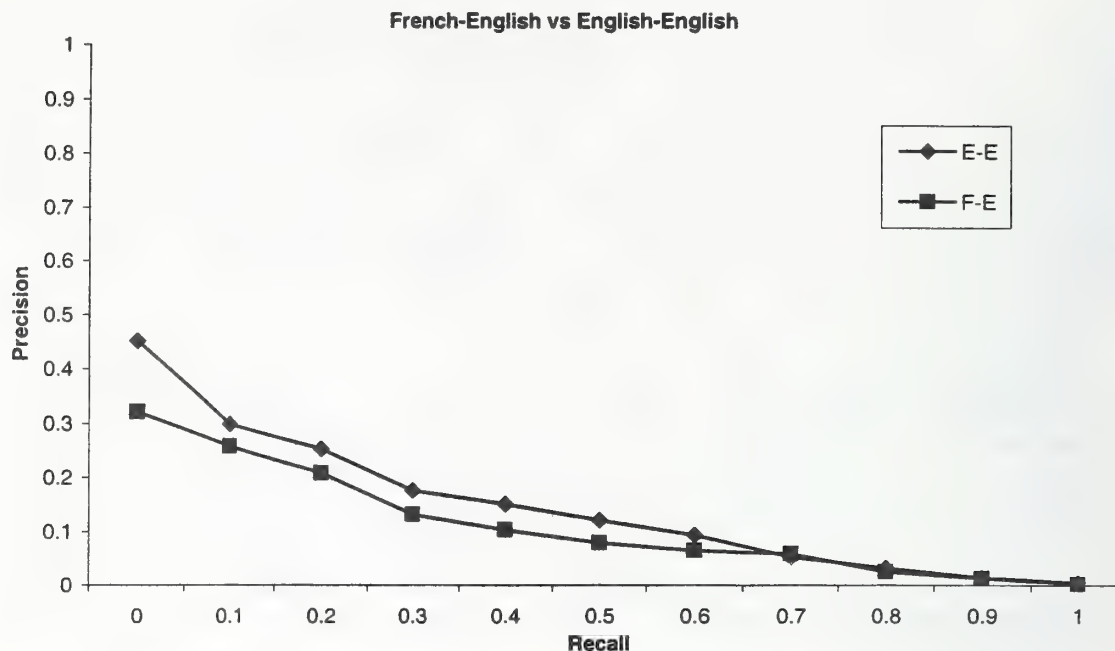


Figure 4: Cross-language versus Monolingual: English documents.

A similar comparison of English document retrieval in response to French queries versus the monolingual case where equivalent English queries are used, illustrated in Figure 4, indicates that this cross-language retrieval performance is also at 75% of monolingual performance. The French-English run has an average precision of 0.1010 versus an average precision of 0.1331 for the monolingual English-English run. These results show nice consistency, though not quite at the level of performance we would have hoped for.

4. Retrieval Performance.

Although the CINDOR system, as evaluated so far, has demonstrated performance improvements over a baseline system without the conceptual interlingua and proper name recognition modules, and has performed consistently at 75% of monolingual precision in both directions in the English and French language pair, the system consistently under performs against comparable systems in terms of average precision. This is obvious from examination of both TREC-7 and TREC-8 results, as well as other experiments we have undertaken, including experiments in other language combinations.

Since the low average precision scores have been in evidence in all experiments we have conducted, using many different configurations both in terms of system set-up and test environment (document collection, queries, language pairs, etc), it seems likely that the problem is inherent in the retrieval algorithm being used and is not due to mis-translations of terms from particular queries. Unfortunately, given the current architecture of the system, we are reliant

upon the retrieval mechanism of the Oracle ConText system and are unable to determine exactly the parameters of the retrieval function being used.

We have however undertaken a standard benchmark comparison against the CINDOR system and the SMART retrieval system, which has been used in extensive experimentation on term weighting algorithms [Salton & Buckley 1997]. The objective of this experiment is to test our hypothesis that the ranking algorithm is impacting our performance. A particularly useful feature of the SMART system in this case is its ability to use a range of different ranking functions which can be specified quite simply using a standard notation. Our initial hypothesis is that the ConText system provides a straightforward *tf*idf* ranking of documents, denoted as *ntn.ntn* in SMART notation. The final CINDOR ranking, which applies pivoted length normalization in order to re-rank the initial results, therefore equates to *ntu.ntu* in SMART notation.

We tested our hypothesis using the collection of 243,000 Associated Press documents used in the TREC cross-language track and the 28 TREC-7 cross-language track queries in English, for which we already had results using CINDOR. This is a simple monolingual English experiment for the purpose only of establishing the performance of our hypothesized CINDOR retrieval function versus one which has been shown to perform well over TREC data in the SMART system (*Lnu.ltu*). The results of our experiment are presented in Table 1 below.

These results confirm our hypothesis, that CINDOR's final ranking is equivalent to a simple *ntu.ntu* function, while the SMART *Lnu.ltu* formula (Singhal *et al* 1996) provides substantially better performance – 80% better in this case. This is in fact only a subset of a range of detailed ranking experiments we have conducted, which confirm the importance of the retrieval weighting function in our overall results, cross-language or otherwise.

	Average Precision
CINDOR	0.2515
SMART	
<i>ntu.ntu</i>	0.2426
<i>Lnu.ltu</i>	0.4531

Table 1: CINDOR ranking versus SMART variants.
(Associated Press collection – TREC-7 CLIR English queries)

This conclusion is of course not news, especially in the context of the *eighth* text retrieval conference, but this straightforward investigation has served to succinctly pinpoint the problem which has resulted in the CINDOR system comparing poorly to published results in a range of experiments in various language combinations. More importantly, it points immediately to the solution of this problem and indicates the direction in which our work should proceed. It is critical that we put the CINDOR system on equal footing with other systems in terms of the retrieval function used so that we can then more clearly establish the advantages and disadvantages of our conceptual interlingua approach to cross-language retrieval.

5. Conclusion.

Our TREC-8 experiments reported here are part of a wider and ongoing series of evaluation experiments designed to establish the performance of the CINDOR retrieval system over a range of language combinations and text types, and more broadly to evaluate the usefulness of our conceptual interlingua approach to cross-language information retrieval.

The results presented here for English and French suggest that there are benefits to be had from the use of our conceptual interlingua resource. In comparing the CINDOR system against a simple baseline for monolingual retrieval, and against that baseline system using Systran machine translation of queries for cross-language English-French retrieval, CINDOR provided 40-45% gains in average precision over the baseline system. We have also established a consistent level of cross-language performance using the CINDOR system, when compared to equivalent searches in a monolingual environment using same-language queries and documents. In both English-French and French-English, average precision in cross-language searches was measured at 75% of the level achieved in equivalent monolingual experiments.

We have noted however, that although these comparative results between different experiments with the CINDOR system are informative, the low absolute level of precision achieved using CINDOR across a range of experiments is an impediment to useful comparisons between our conceptual interlingua approach to cross-language retrieval and other approaches which have been tried and evaluated in the TREC cross-language track and elsewhere. A straightforward investigation using the SMART retrieval system was enough to verify our hypothesis that the low level of performance was attributable to an overly simplistic retrieval function, and that replacement of this algorithm with a state-of-the-art weighting scheme could deliver on the order of 80% improvement in average precision. Addressing the retrieval weighting function problem is therefore an important component of our future work.

References:

[Diekema *et al* 1999]

Diekema A, Oroumchian F, Sheridan P, and Liddy E, "TREC-7 Evaluation of Conceptual Interlingua Document Retrieval (CINDOR) in English and French", *Proceedings of the Seventh Text Retrieval Conference (TREC-7)*, NIST Special Publication, 1999.

[Miller 1990]

Miller G., "WordNet: An On-line Lexical Database", *International Journal of Lexicography*, Vol. 2, No. 4, Special Issue, 1990.

[Salton & Buckley 1997]

Salton G, and Buckley C, "Term-weighting Approaches in Automatic Text Retrieval", In *Readings in Information Retrieval*, K. Spark-Jones & P. Willet (Eds), Morgan Kaufmann, San Francisco, CA, 1997, pages 323-238.

[Singhal *et al* 1996]

Singhal A, Buckley C, and Mitra M, "Pivoted Document Length Normalization", In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, August 1996, ACM Press, pages 21-290.

CLIR using a Probabilistic Translation Model based on Web Documents

Jian-Yun Nie

Laboratoire RALI,

Département d'Informatique et Recherche opérationnelle,

Université de Montréal

C.P. 6128, succursale Centre-ville

Montréal, Québec, H3C 3J7 Canada

nie@iro.umontreal.ca

In this report, we describe the approach we used in TREC-8 Cross-Language IR (CLIR) track. The approach is based on probabilistic translation models estimated from two parallel training corpora: one established manually, and the other built automatically with the documents mined from the Web. We describe the principle of model building, the mining of parallel texts, as well as some preliminary evaluations.

1. Introduction

Last year, in TREC7, we compared three possible approaches to CLIR (for French and English), namely, the approach based on a bilingual dictionary, the approach based on a machine translation (MT) system, and the approach based on a probabilistic translation model using parallel texts. It has been shown that the dictionary-based approach did not give satisfactory performance. The approach using an MT system gave a good performance. In the case of the probabilistic model, the performance was close to that of MT approach.

In TREC7, the IBM group [Franz98] used a similar approach, but for document translation (instead of query translation as in our case) and using long queries (instead of short queries in our case for TREC7). Their system was one of the bests in TREC7 CLIR runs. This is an encouraging result that shows the approach based on a probabilistic model may perform very well.

In TREC8, our goal is to continue using our approach based on parallel texts, but we want to test the performance of a probabilistic model that is estimated from a set of parallel texts automatically mined from the Web. The purpose of these tests is to see if automatic mining of parallel texts may be a possible solution to the problem of unavailability of parallel texts for several language pairs. For the moment, we only have a model estimated for English-French. So our submitted runs only concern English and French documents (AP and SDA collections) using either English or French queries. Two sets of runs have been submitted: one with a probabilistic model trained with a manually established corpus - the Hansard; and the other with a model trained by the Web texts.

In the following sections, we will first recall the principle of building a probabilistic translation model from parallel texts. Then we will describe briefly the way in which parallel texts are mined from the Web. Finally we will give a description of some experimental results.

2. Principle of building a probabilistic translation model

Given a set of parallel texts in two languages, they are first aligned into parallel sentences. The criteria used in sentence alignment are the position of the sentence in the text (parallel sentences have similar positions in two parallel texts), the length of the sentence (they are also similar in length), and so on [Gale93]. In [Simard92], it is proposed that cognates may be used as an additional criterion. Cognates refers to the words (e.g. proper names) or symbols (e.g. numbers) that are identical (or very similar in form) in two languages. If two sentences contain such cognates, it provides additional evidence that they are parallel. It has been shown that the approach using cognates performs better than the one without cognates.

Once a set of parallel sentences is obtained, word translation relations are estimated. First, it is assumed that every word in a sentence may be the translation of every word in its parallel sentence. Therefore, the more two words appear often in parallel sentences, the more they are thought of to be translation of one another. In this way, we obtain some initial probabilities of word translation.

At the second step, the probabilities are submitted to a process of Expectation Maximization (EM) in order to maximize the probabilities with respect to the given parallel sentences. The algorithm of EM is described in [Brown93]. The final result is a probability function $P(f|e)$ which gives the probability of f to be the translation of e . Using this function, we can determine a set of probable word translations in the target language for a query in the source language.

3. Mining parallel texts from the Web

The problem we often have with probabilistic models is the unavailability of parallel texts for many language pairs. The Hansard corpus is one of the only existing corpora for English and French. For other languages (e.g. Chinese and English), such a corpus is less (or not at all) available. In order to solve this problem, we conducted a text-mining project in the Web in order to find parallel texts automatically. The first experiments with the mined documents have been described in [Nie99]. The experiments were done with a subset (5000) of the mined documents. However, they showed that the approach is feasible. In TREC8, we intend to evaluate the performance of a probabilistic model trained with all the parallel documents we found (about 20 000 pairs).

The mining process is devised into several steps:

- selection of candidate web sites
- finding all the documents from the candidate sites
- paring the texts using simple or sophisticated criteria

The first step aims to determine the possible web sites where there may be parallel texts for the given language pair. The way we did this is to send requests to some search engines, asking for French documents containing an anchor named "English version", "english", and so on; and similarly for English documents. The idea is, if a French document contains such an anchor, the link to which the anchor is associated usually points to the parallel text in English.

From the set of documents returned by the search engines, we extract the addresses of web sites, which are considered as candidate sites.

The second step also uses the search engines. In this step, a series of requests are sent to the search engines to obtain the URLs of all the documents in each site.

The last step consists of paring up the URLs. We used some heuristic rules to determine quickly if an URL may be parallel to another:

- First, parallel texts usually have similar URLs. The only difference between them is often a segment denoting the language of the document. For example, "-en", "-e", and so on for English documents. Their corresponding segments for French are "-fr", "-f", and so on. Therefore, by examining the URLs of the documents, we can quickly determine which files may be a pair.
- We then use other criteria such as the length of the file to further confirm or reject a pair.
- The above criteria do not require downloading the files actually. Once a set of possible pairs is determined, the paired files are downloaded. Then we can perform some checking of the document contents. For example, are their HTML structures similar? Do they contain enough text? Can we align them into parallel sentences?

The above process was launched and stopped after 75 hours. We obtained about 20 000 pairs that amount to 135 Mbytes French texts and 118 Mbytes English texts. It is to be noticed that only 30% of 5474 candidate sites have been explored.

4. Experiments

We used a modified version of SMART system [Buckley85] for monolingual document indexing and retrieval. The *ltn* weighting scheme is used for documents. For queries, we used the

probabilities provided by the probabilistic model, multiplied by the *idf* factor. From the translation words obtained, we retained the top *n* words. The value of *n* is determined using TREC6 and TREC7 data.

4.1. Tests with TREC6 and TREC7 data

The purpose is to determine the optimal value of *n* (the number of translation words kept for each query) for each direction (E to F or F to E) and each model. The test runs gave the following performances (measured in average precision) using the long queries:

English to French

n	Hansard		Web	
	TREC6	TREC7	TREC6	TREC7
10	0.2745	0.2685	0.2642	0.2554
15	0.2842	0.3102	0.3193	0.2641
20	0.2861	0.3215	0.3146	0.2918
25	0.2932	0.3184	0.3160	0.2963
30	0.2930	0.3193	0.3242	0.3043
35	0.2930	0.3219	0.3239	0.3076
40	0.2932	0.3241	0.3242	0.3076
45	0.2937	0.3238	0.3258	0.3078
50	0.2938	0.3246	0.3277	0.3083
60	0.2950	0.3249	0.3278	0.3124
70	0.2943	0.3248	0.3288	0.3125
80	0.2894	0.3244	0.3279	0.3124
90	0.2893	0.3238	0.3279	0.3131
100	0.2900	0.3242	0.3274	0.3127

French to English

n	Hansard		Web	
	TREC6	TREC7	TREC6	TREC7
10	0.2675	0.3855	0.2857	0.3584
15	0.2959	0.3879	0.2992	0.3606
20	0.2944	0.3898	0.3047	0.3665
25	0.2943	0.3918	0.3105	0.3721
30	0.2936	0.3978	0.3102	0.3732
35	0.2929	0.3721	0.3095	0.3738
40	0.2929	0.3699	0.3099	0.3741
45	0.2884	0.3666	0.3097	0.3746
50	0.2690	0.3669	0.3086	0.3740
60	0.2697	0.3371	0.3089	0.3744
70	0.2696	0.3250	0.3097	0.3748
80	0.2696	0.2987	0.3097	0.3743
90	0.2692	0.2982	0.3092	0.3744
100	0.2688	0.2981	0.3090	0.3742

Fig. 1. Tests of the models on TREC6 and TREC7

As we can see in these tables, in the case of the Hansard model, the optimal number of translation words is 60 for English to French translation, and about 30 for French to English translation. In the case of the Web model, the number of 70 seems to be quite good for all the cases. Therefore, these numbers have been chosen.

Each translated query, a list of weighted words, is further transformed by the mtn weighting scheme of SMART. It is then run against the documents of the target language. In parallel, the documents in the target language are retrieved using the original query in the target language. The two sets of results are merged and ordered according to their similarity with the queries. The following four runs have been submitted (all for only English AP and French SDA collections):

RaliHanE2EF: Using English queries and the Hansard model
 RaliHanF2EF: Using French queries and the Hansard model
 RaliWebE2EF: Using English queries and the Web model
 RaliWebF2EF: Using French queries and the Web model

What we can also observe in the above table is that the Web model performs generally slightly better than the Hansard model. In [Nie99], with the limited web model trained with 5000 pairs of parallel texts, the performance was not as good as that of the Hansard model. The above tables show that with enough parallel texts from the Web (actually about the same volume of texts as in the Hansard), we can do as well as with a well controlled parallel corpus.

4.2. Evaluation of the submitted runs

From the official evaluation, we extracted those for AP and SDA collections. We use this set of judgements as our reference. The following table gives the average precision for each run.

E2EF		F2EF	
Hansard	Web	Hansard	Web
0.3027	0.2744	0.3002	0.3012

Fig 2. Merged CLIR tests with TREC8 queries

This table shows that the Web model performs slightly worse than the Hansard model in the E2EF case. In F2EF, the performances are equivalent. At this point, several questions may be raised: Why the optimal numbers set for TREC6 and TREC7 do not work well for TREC8? Is this difference due to the different numbers of translation words used in different runs? To the difference between the sets of queries? Or to the merging method used?

These questions can only be answered when we have thoroughly analyzed the translation and retrieval results with different models. This will be reported later.

Notice that the submitted runs do not use a combination of a probabilistic model and a bilingual dictionary. Our previous tests all confirmed that such a combination improve the performances. Our goal in TREC8 is solely to compare the two probabilistic models. Therefore, the possible improving techniques (such as the combination as well as quasi-relevance feedback) that may be used for both models are not used. In so doing, we hope to be able to have a more clear comparison between the models.

In order to evaluate the performances of the translation models, without considering the problem of result merging, we compare the result of simple cross-language results (from English query to French documents, or vice versa) with those of the monolingual runs. The following table shows this comparison.

French mono: 0.3946		English mono: 0.3090	
E2F (% mono)		F2E (% mono)	
Hansard	Web	Hansard	Web
0.3253 (84%)	0.3109 (79%)	0.2842 (92%)	0.2784 (90%)

Fig. 3. Single CLIR runs

Let us look at some of the problems in query translation.

Wrong translations of the key concepts:

Query 59 - exportation of dangerous medicines

We observe a drastic drop in the case of web model (from 0.4062 of monolingual run to 0.0448 only). The reason is the wrong translation of "medicines" as "médecine" (medical area). We obtained the same performance as the monolingual run using the Hansard model.

Query 60 - Rare Birds Stolen

The Hansard model translated "bird" by itself, and attributed the strongest probability to it. This is the main reason of drastic drop of effectiveness: 0.0568 compared to 0.3392 in monolingual run (we obtained 0.1684 with the Web model). By both models, several terms such as "navire" (ship) have been given very strong probabilities. These terms are translations of some less important terms in the English query (e.g. "shipped") in the description field.

Query 71 - saving the dolphin

In the Web model, the word "dolphin" is translated by itself. This topic also contains the word "net" (fishing net). This raised a lot of problem to both translation models. It is translated as "net" (an adjective in French, or Internet).

Wrong proper name

The French word "dauphin" has been translated as "dauphin" by both translation models. In the case of the Hansard model, this is because "Dauphin" is the name of a place. In the Web model, "Dauphin" is also taken in this way. We found many occurrences in the English Web documents talking about "Dauphin Lake Basin", or phone number in "Dauphin".

Unknown words:

In the Query 64, the words "fertilizer" and "fertilizing" are unknown to the Hansard model. Whereas only "fertilizing" is unknown to the Web model. As a consequence, the CLIR run with the Web model (0.2759) is comparable to that of the monolingual run (0.2519), whereas that with the Hansard model is much worse (0.0260).

"ONU" (UN) is an important concept in French query 61 (on "German UN force"). However, its translation "United Nation" is only attributed with low probabilities (especially by the Web model). A possible reason is the very low frequency of occurrences of "ONU" in the training corpus.

For the Web model, the word "Galiciens" is an unknown French word. For the Hansard model, the situation is even worse: "Catalan", "Galice" and "ETA" are also unknown. The performances obtained with the translations are only 1/2 and 1/3 of the monolingual run.

Related words included

In several cases, we observed the interesting phenomenon that related words are also included in the translation. These related words may be even absent in the original queries. For example, the word "movie" does not appear in the English query about "European film industry". In the translations (by moth models) from French to English, it is included and attributed with a strong probability. As a consequence, the translated queries lead to higher effectiveness (around 0.26) than the original English query (0.1544). The same phenomenon is observed for Query 65 (on synthetic fertilizing): From the French query, some related English words (that are absent in the original English query) have been included in the translations (e.g. environment). In this case, the translated queries also lead to higher performance than the monolingual English run.

5. Final remarks

Some of the above mentioned problems may be solved to certain extent by using the translation models in conjunction with a bilingual dictionary. For example, the unknown words problem and the wrong proper name problem. Such a combination has proven to be effective [Nie99].

The other problems (especially the wrong translation problem) seem difficult to solve. However, it is to be noted that the same problem also occurs for query translation with any tool (MT or bilingual dictionary). These problems explain why CLIR effectiveness is usually lower than the monolingual runs, even with the best translation tools of the world. On the other hand, if we compare the probabilistic translation models with other translations means (in particular, with MT systems), their performances are very close [Nie99]. This suggests that probabilistic models are translation tools that are as valuable as MT systems for the CLIR purposes.

Our tests in TREC8 showed that using Web documents to train a probabilistic model is a reasonable approach. The final performance is only slightly lower than using a controlled parallel corpus. The great advantage of this approach is that it may be easily extended to several other

language pairs with little additional cost. We are extending this approach to several other language pairs.

References

- [Brown93] P. F. Brown, S. A. D. Pietra, V. D. J. Pietra, and R. L. Mercer, The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, vol. 19, pp. 263-312 (1993).
- [Gale93] W. A. Gale, K.W. Church, A program for aligning sentences in bilingual corpora, *Computational Linguistics*, 19 :1, 75-102 (1993).
- [Franz98] M. Franz, J.S. McCarley, S. Roukos, Ad hoc and multilingual information retrieval at IBM, *The Seventh Text Retrieval Conference (TREC-7)*, NIST SP 500-242, pp. 157-168 (1998)
- [Nie98] J.Y. Nie, TREC-7 CLIR using a probabilistic translation model, *The Seventh Text Retrieval Conference (TREC-7)*, NIST SP 500-242, pp. 547-553 (1998).
- [Nie99] J.Y. Nie, P. Isabelle, M. Simard, R. Durand, Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the Web, *ACM-SIGIR conference*, Berkeley, CA, pp. 74-81(1999).
- [Simard92] M. Simard, G. Foster, P. Isabelle, Using Cognates to Align Sentences in Parallel Corpora, *Proceedings of the 4th International Conference on Theoretical and Methodological Issues in Machine Translation*, Montreal (1992).

Berkeley's TREC 8 Interactive Track Entry: Cheshire II and Zprise

Ray R. Larson
School of Information Management and Systems
University of California, Berkeley
Berkeley, CA 94720-4600
ray@sherlock.berkeley.edu

Abstract

This paper briefly discusses the UC Berkeley entry in the TREC8 Interactive Track. In this year's study twelve searchers conducted six searches each, half on the Cheshire II system and the other half on the Zprise system, for a total of 72 searches. Questionnaires were administered to each participant to gather information about basic demographic and searching experience, about each search, about each of the systems, and finally, about the user's perceptions of the systems. In this paper I will briefly describe the systems used in the study and how they differ in design goals and implementation. The results of the interactive track evaluations and the information derived from the questionnaires are then discussed and future improvements to the Cheshire II system are considered.

Introduction

The primary goals of UC Berkeley entry in the TREC-8 Interactive track were to 1) attempt to replicate our entry in the TREC-6 and TREC-7 Interactive track with a larger number of participants (searchers), and 2) to evaluate changes to the experiment system (Cheshire II) to see if there were substantial differences in the ranking of the systems between previous year's entries and this year. In addition we are continuing to use the same systems, questionnaires, and complete TREC-7 Interactive track protocol to obtain further information that we hope to combine with the data obtained in previous TREC interactive track experiments for further analysis.

In TREC-8 we used virtually identical implementations of the Cheshire II system and the ZPRISE system as those used in previous TRECs. The database and indexing for each system were also the same as for TREC-6 and TREC-7 (Larson & McDonough, . The changes made to the Cheshire II system for this year's experiment are discussed below.

The Cheshire II System

The design and retrieval algorithm of the Cheshire II system have been discussed in both the TREC-6 and TREC-7 papers, and only the highlights of that description will be repeated here. The Cheshire II system finds its primary usage in full text or structured metadata collections based on SGML and XML, often as the search engine behind a variety of WWW-based "search pages" or as a Z39.50 server for particular applications. The Cheshire II system includes the following features:

1. It supports SGML and XML as the primary database format of the underlying search engine
2. It is a client/server application where the interfaces (clients) communicate with the search engine (server) using the Z39.50 v.3 Information Retrieval Protocol.
3. It includes a programmable graphical direct manipulation interface under X on Unix and NT. There is also CGI interpreter version that combines client and server capabilities.
4. It permits users to enter natural language queries and these may be combined with Boolean logic for users who wish to use it.

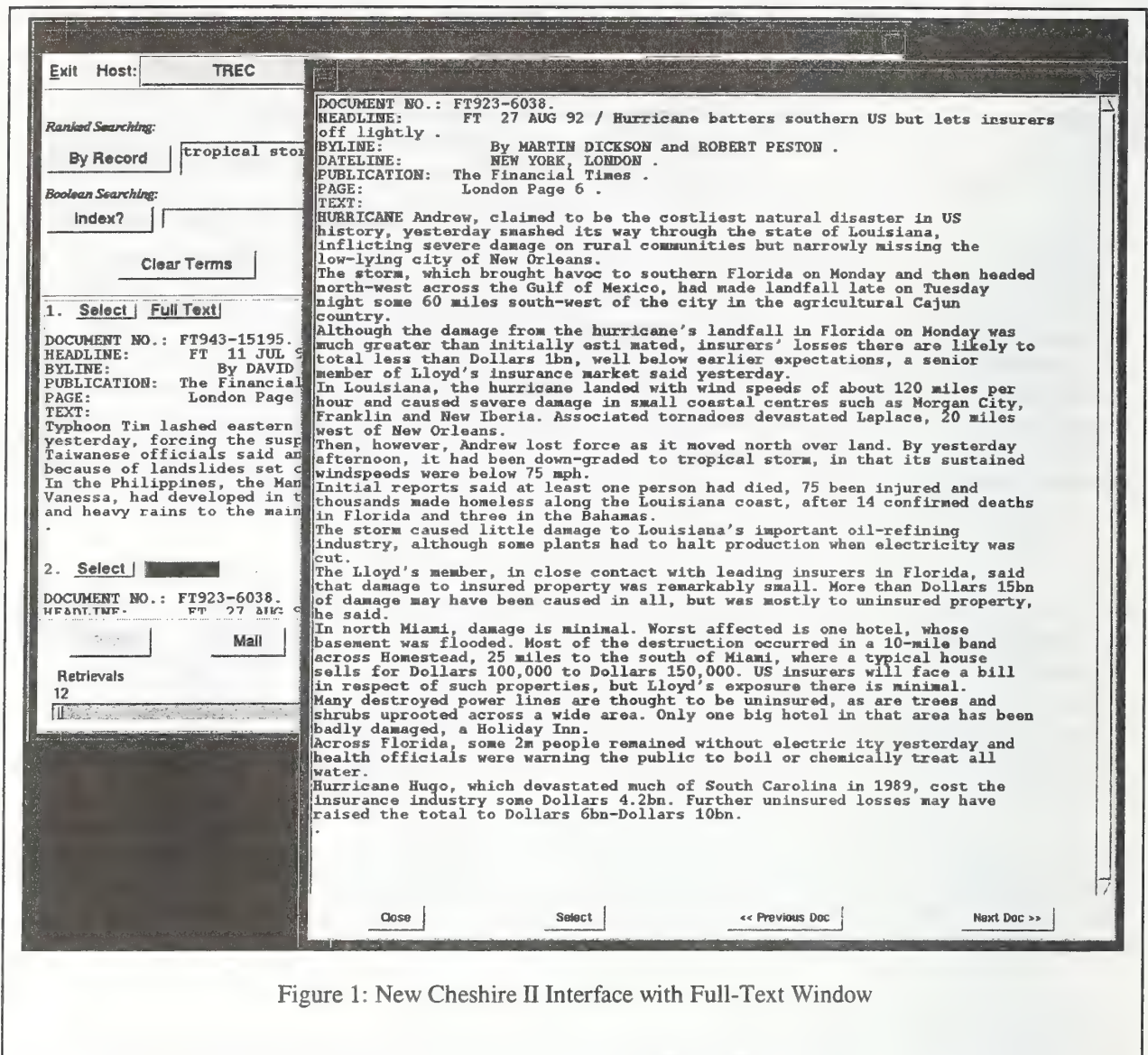


Figure 1: New Cheshire II Interface with Full-Text Window

5. It uses probabilistic ranking methods based on the Logistic Regression research carried out at Berkeley to match the user's initial query with documents in the database.
6. It supports open-ended, exploratory browsing through following dynamically established linkages between records in the database, in order to retrieve materials related to those already found. These can be dynamically generated "hypersearches" that let users issue a Boolean query with a mouse click to find all items that share some field with a displayed record.
7. It uses the user's selection of relevant citations to refine the initial search statement and automatically construct new search statements for relevance feedback searching.

The Cheshire II search engine supports both probabilistic and Boolean searching. The design rationale and features of the Cheshire II search engine have been discussed in the TREC-6 and TREC-7 papers (Larson & McDonough, 1998; Gey, Jiang, Chen & Larson, 1999).

The Cheshire search engine functions as a Z39.50 information retrieval protocol server providing access to a set of databases. In the TREC-8 experiments the TREC Financial Times (FT) database was the only database used by participants. The system supports various methods for translating a searcher's query into the terms used in indexing the database. These methods include elimination of unused words using field-specific stopword lists, particular field-specific query-to-key conversion or "normalization" functions, standard stemming algorithms (Porter stemmer).

The Cheshire II search engine supports both Boolean and probabilistic searching on any indexed element of the database. In probabilistic searching, a natural language query can be used to retrieve the records that are estimated to have the highest probability of being relevant given the user's query. The search engine supports a simple form of relevance feedback, where any items found in an initial search (Boolean or probabilistic) can be selected and used as queries in a relevance feedback search.

The probabilistic retrieval algorithm used in the Cheshire II search engine is based on the *logistic regression* algorithms developed by Berkeley researchers (Cooper, et al. 1992, 1994a, 1994b). The Cheshire II search engine also supports complete Boolean operations on indexed elements in the database, and supports searches that combine probabilistic and Boolean elements.

Relevance feedback is supported and implemented quite simply, as probabilistic retrieval based on extraction of content-bearing elements (such as titles, subject headings, etc.) from any items that have already been seen and selected by a user. At the present time we do not use any methods for eliminating poor search terms from the selected records, nor special enhancements for terms common between multiple selected records (Salton & Buckley 1990).

The Cheshire II Client Interface

The design of the Cheshire II client interface (shown with the TREC FT database in Figure 1), has also been discussed in previous TREC papers. This discussion will concentrate on changes made to the interface for the purposes of our TREC-8 experiment. The Cheshire II interface was intended to provide a generic interface to Z39.50 servers, primarily for search and display of library catalog information and other bibliographic databases. The principle design goals in the interface design were:

1. to support a consistent interface to a wide variety of Z39.50 servers, and to dynamically adapt to the particular server.
2. to reduce the cognitive load on the users wishing to interact with multiple distributed information retrieval systems by providing a single interface for them all.
3. to minimize use of additional windows during users' interactions with the client in order to allow them to concentrate on formulating queries and evaluating the results, and not expend additional mental effort and time switching their focus of attention from the search interface to display clients;

As pointed out in the TREC-7 paper (Gey, Jiang, Chen & Larson, 1999), the interface design assumed that most of the information retrieved and viewed in the search interface would be brief metadata records for documents, and not full text documents themselves. The ability to view full-text documents such as the FT articles used in the interactive track experiments was initially added to the existing interface as longer records that could be scrolled in the main display window. However, comments and questionnaire responses from TREC-7 participants indicated that the separate document viewing window associated with the ZPRISE system was preferable to having to do so much scrolling to accomplish the Interactive Track tasks.. The primary addition to the Cheshire II client interface was the addition of a full-text display window that included controls for selecting/saving the displayed document. This window is shown in Figure 1. The full-text window is invoked by the "Full Text" button next to the "Select" button for each record. The "Full

Text” button changes color to indicate the currently displayed full-text document (blue) or previously seen documents (orange/gold). The full-text window also included controls for stepping directly to the next or previous full-text document in the retrieval list.

In addition, the Boolean NOT, requested by several searchers in TREC-7 was brought out to the interface and integrated with the Boolean search capability.

The Zprise System

The second (control) system used in the TREC-7 Interactive track at Berkeley was the Zprise system from NIST. This system was used in the same configuration and with the same database indexing setup as used for the global control system in our TREC-6 and TREC-7 Interactive Track entries. Zprise, as configured for this test was limited to a total of 24 retrieved items and relevance feedback was disabled. However, the interface was set up so that it provided a very good fit for the tasks involved in the interactive track. For example, documents were viewed in full text form in a separate window from the short display (consisting primarily of title and date as well as control elements for indicating relevant documents and for moving around in the brief display. Most of our users found the ZPRISE displays simple to learn and to operate, in fact most found that the operations required to carry out the Interactive Track tasks were easier to do on the ZPRISE interface than they were on the Cheshire II interface. This was not entirely surprising, since the ZPRISE interface is designed to support TREC-like databases containing full text. We had hoped that the addition of the full-text display to the Cheshire II system would show less difference in preference (and hopefully, less differences in the aspectual recall and precision figures) when compared to TREC-7. But, as discussed below, this hope was not fulfilled.

TREC Interactive Track

The administration of the Interactive Track followed the protocols set down in the track guidelines. This mandated a minimum group of 12 participant searchers, each of whom conduct 6 searches, half on the control system (ZPRISE, identified as “Z”) and half on the experimental system (Cheshire II, identified as “C”). Each searcher was asked to use the features of the respective interfaces to select as relevant those documents that they considered to relevant to one or more aspects of the specific topic.

System	Data	Topic						Overall Average
		408i	414i	428i	431i	438i	446i	
C	Average of Recall	0.39	0.72	0.37	0.37	0.21	0.24	0.38
	Average of Precision	0.80	0.61	0.77	0.84	0.67	0.44	0.69
Z	Average of Recall	0.42	0.72	0.42	0.40	0.21	0.29	0.41
	Average of Precision	0.93	0.59	0.68	0.86	0.86	0.75	0.78

Table 1. Average Precision and Recall by Topic for Cheshire and Zprise

The pooled results for all systems were evaluated at NIST by the TREC evaluators and “Aspectual Precision” and “Aspectual Recall” for each searcher was calculated. Table 1 shows the values for Aspectual Precision and Recall by TREC topic for the two Berkeley systems (“C” and “Z”, the Cheshire II system and ZPRISE systems respectively) are shown in boldface in Tables 1 and 2. The control system “Z” performed

considerably better than the experimental system in terms of the Aspectual Precision and noticeably better in terms of Aspectual recall. Needless to say, this is a disappointing result, and our analysis has yet to reveal any obvious reason for the discrepancy. We believe that the difference may be due to the more complex interactions required to perform the search tasks on the generic Cheshire II interface than on the ZPRISE system, certainly the comments of participants on the questionnaires indicated that most of them preferred the ZPRISE system.

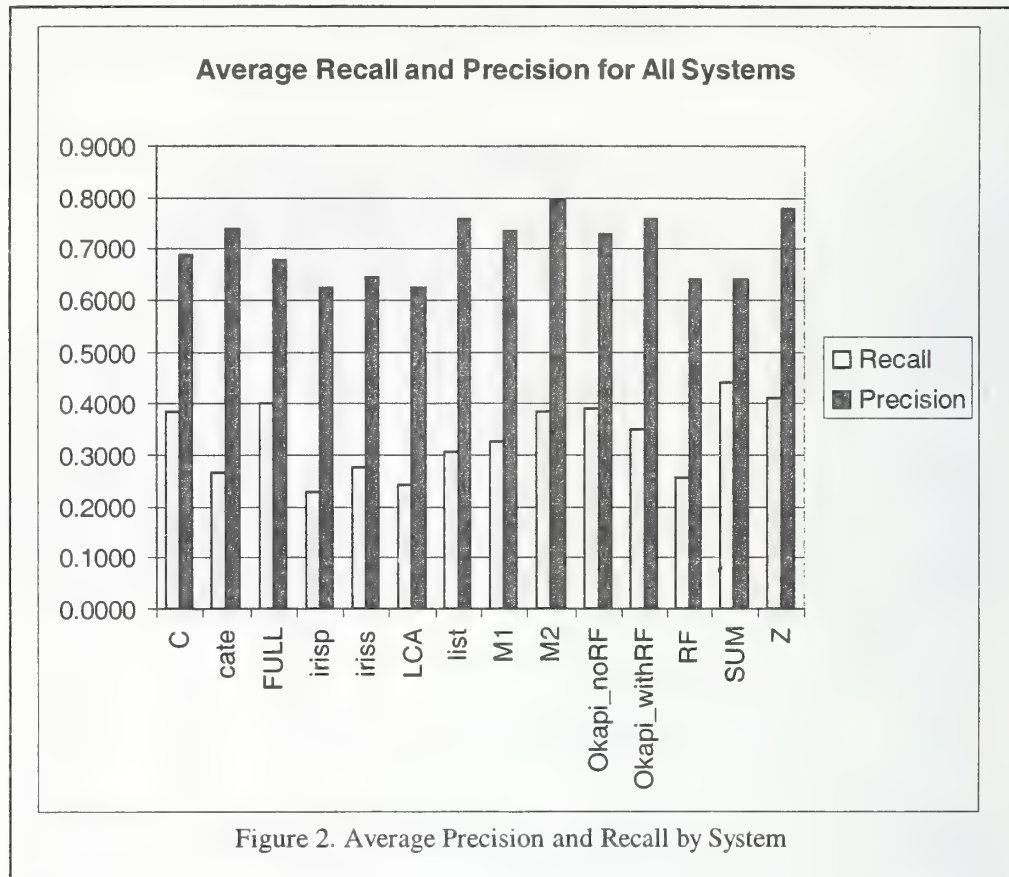
Searcher	Data	System		Mean
		C	Z	
P1	Average of easy_start	3.6667	4.6667	4.1667
	Average of easy_search	4.3333	4.3333	4.3333
P10	Average of easy_start	5.0000	4.6667	4.8333
	Average of easy_search	4.6667	4.6667	4.6667
P11	Average of easy_start	3.6667	3.3333	3.5000
	Average of easy_search	3.6667	3.6667	3.6667
P12	Average of easy_start	3.0000	3.6667	3.3333
	Average of easy_search	3.6667	3.6667	3.6667
P2	Average of easy_start	3.0000	3.6667	3.3333
	Average of easy_search	2.3333	3.0000	2.6667
P3	Average of easy_start	3.3333	3.0000	3.1667
	Average of easy_search	3.6667	3.3333	3.5000
P4	Average of easy_start	3.0000	3.6667	3.3333
	Average of easy_search	3.0000	3.3333	3.1667
P5	Average of easy_start	2.3333	3.6667	3.0000
	Average of easy_search	2.0000	3.0000	2.5000
P6	Average of easy_start	3.0000	3.3333	3.1667
	Average of easy_search	3.0000	3.3333	3.1667
P7	Average of easy_start	4.0000	4.6667	4.3333
	Average of easy_search	4.0000	4.3333	4.1667
P8	Average of easy_start	3.3333	4.0000	3.6667
	Average of easy_search	3.3333	4.3333	3.8333
P9	Average of easy_start	4.0000	4.0000	4.0000
	Average of easy_search	3.6667	4.0000	3.8333
Mean of "easy to start searching"		3.4444	3.8611	3.6528
Mean of "easy to search"		3.4444	3.7500	3.5972

Table 3: Average Ease of Starting Search and Ease of Doing Search for each Participant by system

In the following section we will examine the characteristics of the searchers as reported in the questionnaires administered during the experiments. Figure 3 summarizes the average aspectual precision and recall for each of the systems participating in the TREC-7 Interactive Track.

User Characteristics

The administration of the interactive track followed the track guidelines with a single group of 12 participants. While only one of the participants had used either the experimental (Cheshire II) or control



(ZPRISE) systems in searching tasks, some had seen demonstrations of the experimental system. The searchers who participated in the study were volunteers drawn from the School of Information Management and Systems at UC Berkeley (a call for participation was sent to all students and faculty at SIMS and the first 12 volunteers were scheduled for search sessions. A pre-search questionnaire asked each participant about:

1. What high school/college/university degrees/diplomas do have (or expect to have)?
2. What is your occupation?
3. What is your gender?
4. What is your age?
5. Have you participated in previous TREC searching studies?
6. Overall how long have you been doing online searching?
7. Experience with using a point-and-click interface (e.g. Windows, Macintosh)
8. Experience searching on computerized library catalogs either locally or remotely
9. Experience searching on CD-ROM systems
10. Experience searching on commercial online systems (BRS afterdark, Dialog, Lexis-Nexis, etc.)
11. Experience searching on the World Wide Web search services (Alta Vista, Excite, Yahoo, Hotbot, etc.)
12. Experience searching on other systems
13. How often do you conduct a search on any kind of system
14. "I enjoy carrying out information searches"

All of the participants, except one undergraduate, held college degrees (One held a PhD, Three others were PhD students with previous undergraduate and graduate degrees, and the remaining participants were Masters students in the SIMS program). Three of the participants (P1, P2, and P3) had over 8 years of experience in online searching on other systems. As observed last year, once again the most frequently used search systems were the Web search services and the next most frequent were online catalogs. It appears that most recent searchers will be gaining their experience from the WWW and possibly from online library catalogs, and will probably not have experience (or as much experience) with traditional Boolean systems such as Dialog.

Per Search Results

Following each search the participants were given a questionnaire asking:

1. Are you familiar with this topic
2. Was it easy to get started on this search
3. Was it easy to do the search on this topic
4. Are you satisfied with your search results
5. Are you confident that you identified all of the different instances for this topic
6. Did you have enough time to do an effective search.

Table 4 shows the average responses for the "easy to do the search" and "easy to get started on the search" questions by searcher and system. As may be seen from the table, many searchers found the search easier to do with the ZPRISE system than with the Cheshire II system. Similarly, Table 5 shows the average responses to the "Are you satisfied with the results" question. Here, the overall scores rate the searches done with the Cheshire II system slightly higher than for ZPRISE. Table 6 shows the average responses to the question "Are you familiar with this topic?" Here the responses show that the searchers were generally less familiar with the topics searched on the Cheshire system versus those on the ZPRISE system. Correlation analysis showed, however, no significant correlation between familiarity with a topic and either the ease of searching or the satisfaction with search results.

Post-System Questions

The searches were conducted in blocks of 4 questions on each system. Following the searcher's interaction with a system, a post-system questionnaire was administered. This post-system questionnaire asked each

Average of satisfied	System		
Searcher	C	Z	Mean
P1	4.6667	4.3333	4.5000
P10	4.3333	3.6667	4.0000
P11	3.3333	3.0000	3.1667
P12	3.0000	3.0000	3.0000
P2	3.3333	2.3333	2.8333
P3	2.6667	2.3333	2.5000
P4	2.6667	2.6667	2.6667
P5	2.6667	3.0000	2.8333
P6	3.0000	3.3333	3.1667
P7	3.0000	3.3333	3.1667
P8	3.6667	4.6667	4.1667
P9	3.6667	4.0000	3.8333
Overall means	3.3333	3.3056	3.3194

Table 5: Average User Satisfaction with Search

Average of familiar	System		
Searcher	C	Z	Mean
P1	1.6667	2.0000	1.8333
P10	1.6667	3.6667	2.6667
P11	1.0000	1.3333	1.1667
P12	2.0000	1.6667	1.8333
P2	1.0000	1.6667	1.3333
P3	2.6667	2.0000	2.3333
P4	2.3333	1.6667	2.0000
P5	2.0000	2.3333	2.1667
P6	2.3333	2.3333	2.3333
P7	2.0000	3.0000	2.5000
P8	1.3333	2.6667	2.0000
P9	4.0000	4.0000	4.0000
Overall means	2.0000	2.3611	2.1806

Table 6: Average User Familiarity with Topics

searcher the following questions:

1. How easy was it to *learn to use* this information system?
2. How easy was it to *use* this information system?
3. How well did you *understand how to use* the information system?
4. Write down any comments that you have about your searching experience with this information retrieval system.

Overall, the searchers found both systems very easy to learn. The Cheshire system was marked down again on the “easy to use” question. From the comments, this appeared to be related to some features being hard to understand and use. Some searchers mentioned that it was hard to figure out when and if the items they selected as relevant had been seen before, and as previously observed, the need to scroll back to the beginning of a record to select it as relevant (for those NOT using the full-text window) was a problem when the full text is displayed in the main window.

Exit Questionnaire

After the completion of all searches an exit questionnaire was administered to the searchers. This questionnaire asked:

1. To what extent did you understand the nature of the searching task?
2. To what extent did you find this task similar to other searching tasks that you typically perform?
3. How different did you find the systems from one another?
4. Please rank the two systems in order of how *easy they were to learn to use*.
5. Please rank the two systems in order of how *easy they were to use*.
6. Please rank the two systems in the order of *which system you liked best*.
7. What did you like about each of the systems.
8. What did you dislike about each of the systems.
9. Please list any other comments that you have about your overall search experience.

The searchers claimed to have a very good understanding of the search task (mean was 4.16), and they found the task similar to other searching tasks (mean of 3.50). They also found the systems somewhat different (mean of 3.41). In ranking the systems, 7 out of 12 ranked Cheshire II as easier to learn to use, but only 5 out of 12 ranked it as easier to use. 7 out of the 12 searchers “liked” Cheshire the best of the two systems. However, as the Precision and Recall results show, they did not perform as well using the Cheshire system as they did using ZPRISE. One had a strong preference for the ZPRISE system, but commented that he might have preferred Cheshire if it had been introduced first.

Conclusions

It is very difficult to draw any firm conclusions from the analysis that we have conducted. There is no clear evidence why the Cheshire II system has shown poorer Precision and Recall performance than the control system. One tentative thought is that Cheshire II is providing *too much* functionality and may be confusing the users with too many options. Many of the users did use the Boolean features of the system, and this might have caused a significant reduction in Recall compared to the ranked retrieval offered by the ZPRISE system. These tentative hypotheses will need further analysis to discover if they are supported by the data collected.

Acknowledgements

I would like to thank SIMS PhD students Youngin Kim and Jacek Purat for their much needed help in conducting the user evaluation sessions for this research.

The original development of the Cheshire II system was sponsored by a College Library Technology and Cooperation Grants Program, HEA-IIA, Research and Demonstration Grant #R197D30040 from the U.S. Department of Education. Further development work on the Cheshire II project and system was supported as part of Berkeley's NSF/NASA/ARPA Digital Library Initiative Grant #IRI-9411334. Current work is being supported as part of the "Search Support for Unfamiliar Metadata Vocabularies" research project at UC. Berkeley, sponsored by DARPA contract N66001-97-C-8541; AO# F477. Future development of the Cheshire system is being sponsored by the NSF/JISC International Digital Libraries program.

Bibliography

Cooper, W. S., Gey, F. C., & Dabney, D. P. (1992). Probabilistic Retrieval Based on Staged Logistic Regression. In: *SIGIR '92 (Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, Denmark, June 21-24, 1992)* (pp. 198-210). New York: ACM.

Cooper, W. S., Gey, F. C. & Chen, A. (1994a). Full Text Retrieval based on a Probabilistic Equation with Coefficients fitted by Logistic Regression. In: D. K. Harman (Ed.) *Second Text Retrieval Conference (TREC-2)*, Gaithersburg, MD, USA, 31 Aug.-2 Sept. 1993, NIST-SP 500-215, (pp. 57-66). Washington : NIST.

Cooper, W. S., Chen, A. & Gey, F. C. (1994b). Experiments in the Probabilistic Retrieval of Full Text Documents In: *Text Retrieval Conference (TREC-3) Draft Conference Papers, Gaithersburg, MD : National Institute of Standards and Technology*.

Gey, F. C., Jiang, H., Chen, A. & Larson, R. R. (1999) Manual Queries and Machine Translation in Cross-Language Retrieval and Interactive Retrieval with Cheshire II at TREC-7. In E. Voorhees and D. Harman (Eds.) *Information Technology: The Seventh Text Retrieval Conference (TREC-7)*. NIST special publication 500-242. (pp. 527-540) Gaithersburg, MD : NIST, July 1999.

Larson, R. R. (1991). Classification Clustering, Probabilistic Information Retrieval, and the Online Catalog. *Library Quarterly*, 61, 133-173.

Larson, R. R. (1992). Evaluation of Advanced Retrieval Techniques in an Experimental Online Catalog. *Journal of the American Society for Information Science*, 43, 34-53.

Larson, R. R & McDonough, J (1998) Cheshire II at TREC 6: Interactive Probabilistic Retrieval. In E. Voorhees and D. Harman (Eds.) *Information Technology: The Sixth Text Retrieval Conference (TREC-6)*. NIST special publication 500-240. (pp. 649-649) Gaithersburg, MD : NIST, August 1998.

Ousterhout, J. K. (1994). *Tcl and the Tk Toolkit* Reading, Mass. : Addison-Wesley.

Salton, G. & Buckley, C. (1990). Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science*, 41, 288-297.

TREC-8 Experiments at Maryland: CLIR, QA and Routing

Douglas W. Oard,^{*} Jianqiang Wang,[†] Dekang Lin,[‡] and Ian Soboroff[§]

Abstract

The University of Maryland team participated in four aspects of TREC-8: the ad hoc retrieval task, the main task in the cross-language retrieval (CLIR) track, the question answering track, and the routing task in the filtering track. The CLIR method was based on Pirkola's method for Dictionary-based Query Translation, using freely available dictionaries. Broad-coverage parsing and rule-based matching was used for question answering. Routing was performed using Latent Semantic Indexing in profile space.

1 Introduction

The Eighth Text REtrieval Conference (TREC-8) offered many more attractive evaluation opportunities than our team could have pursued, so we chose to participate in four aspects of the work that are aligned particularly closely with our ongoing work. In Cross-Language Information Retrieval track (CLIR), we focused on rapid retargetability, seeking to learn how well we could do with freely available resources that have more limited vocabulary coverage than those we have used in the past. We also tried out the Inquiry synonym operator as a device for selecting the correct translation, an approach introduced by Pirkola [7] but not previously tested at TREC. In the new Question Answering track, we explored the potential for combining broad-coverage parsing with rule-based matching. Our effort for the Routing task of the Filtering track explored the use of Latent Semantic Indexing on a space formed from profiles that aggregate several documents, in an effort to understand whether common aspects of the topic space could be automatically identified and exploited. Our participation in the Ad Hoc task was limited to a single run with an off-the-shelf retrieval system—as in past years, we used the Ad Hoc task as a learning opportunity for some of the new members of our team while producing results that might help to enrich the assessment pool.

Our team for the first time included significant participation by visitors from other institutions. Dekang Lin from the University of Manitoba worked on Question Answering while on sabbatical at Maryland. Ian Soboroff from the University of Maryland, Baltimore County worked on the Routing task. Our experience suggests that collaborations of this sort can serve the community well, combining fresh ideas with experience that gives a leg up on climbing the learning curve.

2 Cross-Language Information Retrieval

We participated in the main task of the CLIR track, using an English query to create a single merged ranked list of English, French, German and Italian news stories for each of the 28 topics. We sought to answer three questions: (1) what is the best that can be done using freely available resources; (2) how well does Pirkola's method for accommodating multiple candidate translations work on the TREC CLIR collection; and (3) would building a single index be more effective than building separate indices for each language?

A purist approach to the first question would have required that we use a freely available retrieval system such as PRISE, SMART or MG. The second question led us to instead choose Inquiry, which is inexpensively (but not quite freely) available for research use. We downloaded three bilingual "dictionaries," all of which were actually simply lists of English terms that were paired with some equivalent terms in another language.

^{*}College of Library and Information Services, University of Maryland, College Park, MD 20742, oard@glue.umd.edu

[†]College of Library and Information Services, University of Maryland, College Park, MD 20742, wangjq@glue.umd.edu

[‡]Department of Computer Science, University of Manitoba, Winnipeg, R3T2N2 CANADA, lindek@cs.umanitoba.ca

[§]Department of C.S. and E.E., University of Maryland, Baltimore County, Baltimore, MD 21250, ian@cs.umbc.edu

Pair	Source	English Terms	Foreign Terms	Avg Translations
E-G	http://www.quickdic.de	99,357	131,273	1.7
E-F	http://www.freedict.com	20,100	35,008	1.3
E-I	http://www.freedict.com	13,400	17,313	1.3

Table 1: Sources and summary statistics for bilingual dictionaries.

Here we take “terms” to include both single words and multiword expressions—multiword expressions were common in some of the dictionaries. Table 1 shows the source and summary statistics for each dictionary.

Each of the dictionaries was downloaded in a native machine-readable format that was designed for the originally intended use (typically, interactive access using an associated program). No documentation regarding storage formats was provided with any of the dictionaries, but conversion to our standard format turned out to be quite straightforward in every case. We preserved the order of the original dictionary where possible, and an examination of the results indicates that the known translations for each term are stored in lexicographic order. In other work we have reordered the translations by their (unconditioned) frequency in the Brown Corpus (for terms that are present in that corpus) [5], but that was not done in this case.

2.1 Pirkola’s Technique and Multilingual Indexing

Once we had a dictionary in a suitable format, we used it with our existing Dictionary-based Query Translation (DQT) routines to translate the query from English into the language of one of the four language-specific CLIR subcollections (no translation was needed for the English subcollection). In DQT, each query term for which at least one translation is known was replaced with one or more of the known translations. When no translation is known, the English term is retained unchanged in the translated query. Since query terms may have more than one translation, some selection heuristic is needed. In the past we have tried retaining Every Translation (DQT-ET) or just the First Translation (DQT-FT), finding that sometimes one approach yields better average precision and sometimes the other does. We thus elected to try both and to select the best of the two as our baseline for evaluating Pirkola’s technique.

Pirkola used structured queries to attack the problem of translation ambiguity [7]. Specific terms, which are quite useful for searching, typically have relatively few translations. But with DQT-ET, the more translations a query term has, the more weight it will get because every possible translation will appear in the query. With Pirkola’s structured queries, translations of the same term are treated as instances of a single term. In this way, important query terms get relatively more weight. In our experiment, we implemented Pirkola’s technique by grouping all translations for each query term using the Inquiry synonym operator `#syn()`. All of the groups were then combined using Inquiry’s sum operator `#sum()`.¹ Pirkola found that this approach yielded substantial improvements in average precision when compared with an approach similar to DQT-ET.

As in TREC-7, we built a separate index for the documents in each language (English, French, German, and Italian), produced separate ranked lists for each language for each topic using queries translated into only that language, and then applied a uniform merging strategy in which we took n documents from the top of the English list for every 1 document that we took from each other list [6]. In preliminary experiments with TREC-7 data, we found $n = 2$ to be optimal for DQT with these dictionaries. That contrasts markedly with our conclusion at TREC-7 that $n = 10$ was best when queries were translated using a commercial machine translation system. We have not yet investigated this effect in detail, but in the results reported below we use a uniform 2:1:1:1 merge in which each block of 5 documents in the merged list contains 2 English documents, 1 French document, 1 German document, and 1 Italian document.

¹Pirkola also used Inquiry’s `#uw2` operator to group terms in a phrase together. We omitted that from our implementation, so each word in the phrase is treated separately in our runs.

Run ID	Official	Queries	Translation	Index	Merged	English	French	German	Italian
umd99b1	Yes	Long	Pirkola	Monolingual	0.162	0.345	0.113	0.114	0.078
umd99b2	Yes	Long	DQT-FT	Monolingual	0.156	0.345	0.097	0.089	0.062
umd99b3	Yes	Long	DQT-ET	Monolingual	0.134	0.345	0.045	0.071	0.062
umd99c1	Yes	Title	Pirkola	Monolingual	0.100	0.252	0.095	0.066	0.070
umd99c2	Yes	Title	Pirkola	Multilingual	0.103				
umd99c3	No	Title	DQT-ET	Monolingual	0.114	0.252	0.093	0.064	0.068
umd99c4	No	Title	DQT-ET	Multilingual	0.094				
umd99c5	No	Title	DQT-FT	Monolingual	0.097	0.252	0.110	0.059	0.066
umd99c6	No	Title	DQT-FT	Multilingual	0.098				

Table 2: Official and unofficial CLIR runs, overall and by-language average precision.

Good results have also been reported with a unified multilingual index [3], so we also tried that approach. In that case, all documents were indexed together regardless of language, and the translated queries in each language (including the untranslated English queries) were combined on a topic-by-topic basis. The approach results in a single ranked list, so no merging strategy is required. We enabled English stemming for all runs and did not use any stopword lists.

2.2 Results

We submitted five official CLIR runs and scored an additional four unofficial runs locally, as shown in Table 2. Only the “umd99b1” and “umd99c1” runs contributed to the relevance assessment pools. All runs were in the automatic category. Title queries were formed automatically using the words in the title field of each topic description. Long queries were formed using all words in the topic except SGML markup and field titles. Pirkola’s technique clearly outperformed the best DQT technique (DQT-FT) on long queries in every language, achieving a 28% relative improvement in German, 25% in Italian and 16% in French. The differences in German and Italian were found to be statistically significant (at $p < 0.05$ using a two-tailed paired t -test), the difference in French was not ($t = 1.06, p = 0.30$). The difference is less impressive in the merged results, however, achieving only a 4% relative improvement that was not statistically significant ($t = 1.92, p = 0.065$). Figure 1(a) compares the two techniques on a per-query basis, showing that topics for which Pirkola’s technique is better are considerably more common. Pirkola’s technique is quite slow, however, requiring about 8 minutes per long French query on a SPARC 20 (compared with about 1 minute per long French query for either DQT-FT or DQT-ET). We note with some concern that this slowdown occurred with a dictionary in which multiple translations were relatively rare (averaging only 1.3 translations per term).

With title queries, the observed effect is more variable, with Pirkola’s technique performing 12% better relative to DQT-FT in German and 6% better in Italian, but 14% worse in French. With a merged ranked list, Pirkola’s method comes out 3% better than DQT-FT. This is roughly comparable to the 5% better performance of Pirkola’s technique (compared to DQT-FT) when a multilingual index is used. DQT-ET might be the better basis for comparison in this case, since it outperforms DQT-FT on German and Italian (but is again notably worse on French). When the resulting ranked lists were merged, however, DQT-ET produced a dramatic (and as-yet unexplained) improvement. As Figure 1(b) illustrates, the 14% relative improvement over Pirkola’s technique (which is not statistically significant: $t = -1.47, p = 0.15$) is attributable to topics 67, 68, 69, and 79. We examined these topics and the ranked lists, but no obvious explanation was apparent.

We also were not able to find a statistically significant difference between the use of a single multilingual index and our uniform 2:1:1:1 merging strategy for results obtained using separately constructed monolingual indices. We used the multilingual index only with title queries in our experiments. Neither the 3% relative improvement that resulted from multilingual indexing with Pirkola’s technique nor the 2% relative improvement that resulted from monolingual indexing with DQT-FT showed any sign of significance

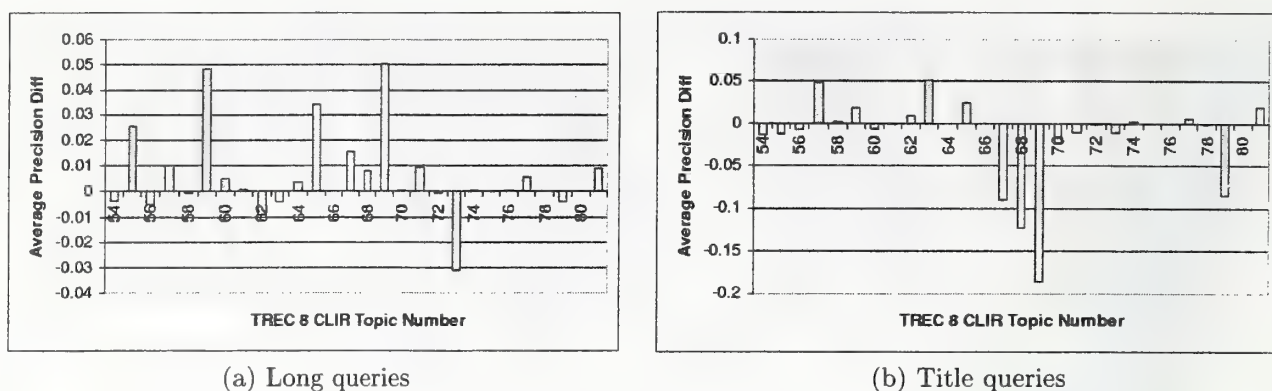


Figure 1: Comparative results by query, merged monolingual. (a) Pirkola's method better above zero, DQT-FT better below. (b) Pirkola's method better above zero, DQT-ET better below.

($t = -0.24, p = 0.81$ and $t = -0.10, p = 0.92$ respectively). The previously unexplained performance of DQT-ET with merged ranked lists produced a 22% relative advantage over multilingual indices, but that difference is also not statistically significant ($t = 1.09, p = 0.28$). Figure 2 shows that again it is topics 67, 69 and 79 that are responsible for the majority of this effect.

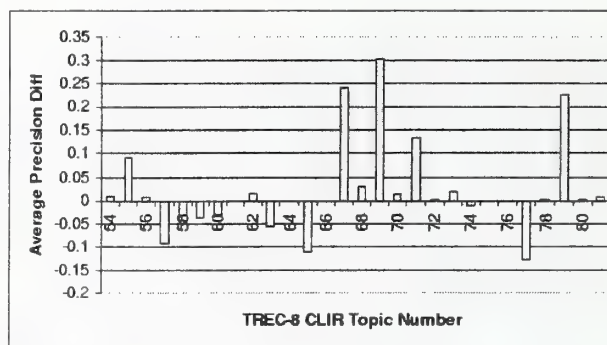


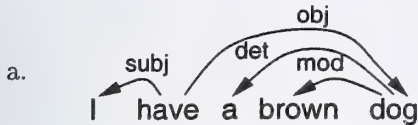
Figure 2: Comparative results by query for DQT-ET on title queries, merging monolingual ranked lists better above zero, single multilingual index better below.

3 Question Answering

Many natural language systems are organized as a stream of processing modules. A parser is usually one of the upstream modules. The resulting parse trees are typically used to guide the processing in downstream modules. For example, a semantic interpreter may rely on the parse trees to identify the atomic components that are semantically interpretable and then combine them according to the parse tree structure to obtain the interpretation for larger chunk of text. We call such processing syntax-guided. A problem with syntax-guided processing is the heavy reliance of the downstream modules on the parse trees. Without the parse trees, a syntax-guided module is usually unable to produce any output. SyncMatcher adopts a syntax-constrained approach where parse trees are used as a source of constraints for downstream modules. Without the constraints, the downstream modules are still functional. The difference is that they will be faced with more ambiguous inputs, which increase the likelihood of error in the output.

The parser used in SyncMatcher is MINIPAR, a principle-based broad-coverage parser. Although MINIPAR uses a constituency grammar internally, its outputs are dependency structures. For each word in the sentence, a dependency structure specifies the governor of the word. For example, (1a) is a dependency structure of a sentence. The root of the dependency tree is “have” and there are 4 dependency relationships in the tree as shown in (1b).

(1)



- b.
- (have subj I)
 - (have obj dog)
 - (dog mod brown)
 - (dog det a)

Given a query and a stream of documents, SyncMatcher matches sentences in the documents against the query using the dependency trees as constraints. Each match is assigned a score, which is used to rank the answers extracted from the documents. The outputs for each query are the top-5 distinct answers.

To find the best match between a query and a sentence in the documents, SyncMatcher first establishes the set of potential correspondence between the words in the query and the words in the documents according to the following rules:

- a word may match another word with identical root form.
- two words match if the result of stemming them with the Porter stemmer is the same.
- A wh-word matches proper nouns that have the same semantic tag as the wh-word. For example, “who” matches named entity that is classified as PERSON.

After collecting the set of potential matching pairs of words, SyncMatcher tries to find a subtrees of the dependency trees of the query and an input sentence that satisfies the following constraints:

(2)

- a. If a node B is on the (undirected) path between two nodes A and C in the dependency tree of the query and A', B' and C' are nodes in the dependency trees of an input sentence that corresponds to A, B and C respectively, then B' must be on the (undirected) path between A' and C' in the dependency tree.
- b. If A' and C' are nodes in the dependency tree of an input sentence and A' and C' corresponds to A and C in the query respectively, there must not exist another node on the path between A' and C' that may also correspond to A or C.

3.1 Semantic Tagging of Wh-words

SyncMatcher answers queries by extracting named entities from the documents. Therefore, we must first determine the type of named entity that the answer belongs to. If the wh-word in the query is “who”, “when”, “where”, “how many” or “how much”, the answer is usually a PERSON, a TIME/DATE, a LOCATION, a NUMBER or an AMOUNT, respectively. When the wh-word in the query is “which”, “what” or “how”, the semantic category of the wh-word is determined by their governor in the dependency tree. For each type of named entity, we constructed a list of common nouns that typically refer to them. For example, the list of common nouns for LOCATION include

country, nation, city, region, republic, island, province, state, town, area, community, territory, capital, world, South, neighborhood, village, land, colony, camp, ...

A wh-word in a query is tagged as type X if its governor belongs to the list of common nouns of the type X. For example, in the query "Which country is Australia's largest export market?", the governor of "which" is "country". Therefore, "which" is tagged as LOCATION. In the query "Which former Ku Klux Klan member won an elected office in the U.S.?", the governor of "which" is "member". Since "member" belongs to a list of words that are very similar to "person", "man", etc., the word "which" is tagged as PERSON.

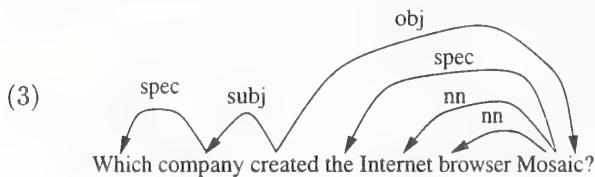
The dependency trees generated by MINIPAR also encodes the following types of coreference relationships: (1) traces and zero pronouns and their antecedents; (2) personal pronouns and their antecedents; and (3) item proper names and their antecedents. The first type coreference relationships are identified during parsing. The other two types are identified by the coreference recognizer borrowed from a University of Manitoba's MUC system.

3.2 A Walkthrough Example

Consider the following query:

Q.108 Which company created the Internet browser Mosaic?

The dependency tree for the query is as follows:

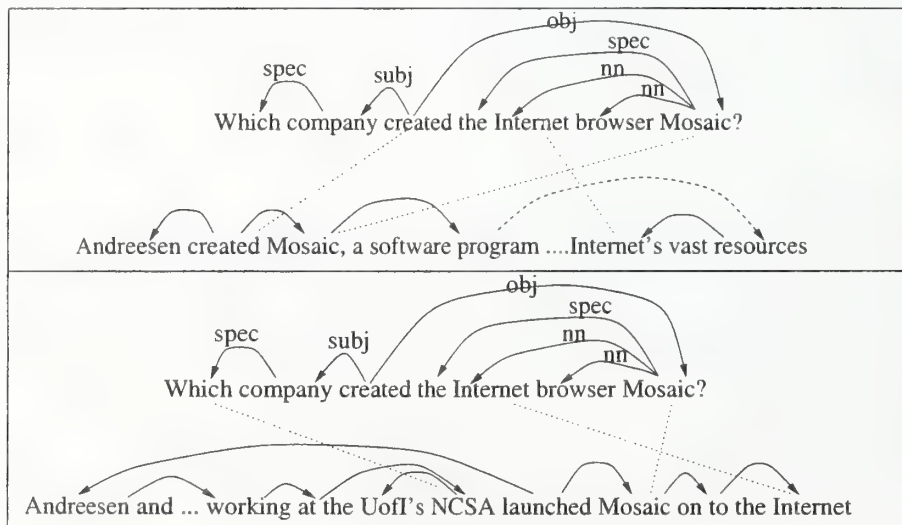


Consider the following fragments from one of the documents:

.... Then he met Marc Andreessen. A 23-year-old cyber-star computer science graduate, Andreessen created Mosaic, a software program that enables even computer novices to explore the Internet's vast resources. Since Andreessen and a group of fellow students working at the University of Illinois' National Center for Supercomputing Applications launched Mosaic on to the Internet last year, it has been used by an estimated 2m people.

The word "Andreessen" is not found in the lexicon in SyncMatcher. However, there is the coreference relationship between "Andreessen" and "Marc Andreessen" earlier in the document. Since "Marc" is a known first name in the lexicon, "Marc Andreessen" is recognized as a person. Therefore, "Andreessen" is tagged as a PERSON. Since the governor of "which" in the query is "company", "which" is tagged as an ORGANIZATION. It can only correspond to words in documents that are also tagged as organizations, such as "University of Illinois" and "National Center for Supercomputing Applications".

SyncMatcher identified the following two matches from the above paragraph.



Both matches involve three words in the query. The second match involves the wh-word in the query and is consequently scored higher. SyncMatcher then returns the matching element for the wh-word, “National Center for Supercomputing Applications” as the answer. The phrase “University of Illinois” also matches “which” in Q.108. However, because “NCSA” is on the path between “University of Illinois” and other matching words, such as “Mosaic”, the constraint (2b) rules it out.

3.3 Experimental Results

We used the documents collected by AT&T Labs using a search engine, which contains 200 documents per query. The total size of the document collection is about 200MB (32M words). The documents are ordered according to the relevance score obtained from the search engine. However, this information is currently ignored. SyncMatcher parsed all the sentences in the documents except those in the headers or footers. The total processing time is about 40 hours on a 233MHz Pentium II with 160MB memory and 6GB disk, running Linux. This is roughly equivalent to 222 words per second.

For 80 out of the 198 questions in the Q&A Track, SyncMatcher returned the correct answer as one of its top 5 answers. The distribution of the answers is shown in the following table.

1st	2nd	3rd	4th	5th	not found
47	14	7	7	5	118

4 Routing

We have been exploring a filtering technique which combines content-based and collaborative aspects [11], and TREC-8 is its first exposure with a large collection. We expected this technique to give some advantage to related families of topics, while not harming performance on other topics.

Since our work has focused on the basic technique and not on adaptation, we only submitted results for routing. While adaptation and profile construction are probably not orthogonal, we hoped that this would help show if our technique works aside from any benefits gained from adaptive filtering.

4.1 Collaborative LSI

We first construct our routing queries using a sophisticated relevance feedback approach. All queries are then collected together, and a latent semantic index (LSI) of the query collection is computed. Test documents are routed in the reduced-dimension LSI space, which should highlight common interests among the queries, and diminish noise. Latent semantic indexing [1] has been used before by Dumais in the TREC Routing task [2]. The key difference in our approach is that we compute the latent semantic index from a collection of queries, rather than a collection of individual documents. Specifically, we collect our routing queries for topics 351-400 into a single term-query matrix, and compute an SVD of this matrix. This should give two advantages over a straightforward application of LSI. First, the LSI space is oriented towards features of the queries, rather than the documents, making it better suited to a routing environment with few saved documents and persistent queries. Second, the LSI space highlights commonalities among queries, so that if queries are similar they can benefit from each other.

In Dumais’ approach, the LSI transformation highlights common features among documents, giving dimensions where groups of documents share co-occurrence patterns of certain weighted terms. This is simply too general, and not related to our problem, which is not to choose among documents but to choose among queries. Hull [4] described a “local LSI” technique, which rather than computing the LSI from the entire collection, computed it from the top n documents in an initial retrieval on the query. This is closer to a query-centric LSI than Dumais, but does not allow for collaboration among queries.

It’s not clear that any collaboration takes place in TREC filtering, since the topics are not necessarily designed to overlap, either in information interest or in actual relevant document sets. However, just from a reading of the topic descriptions, several topics this year seem closely related, as can be seen in figure 3. These groups might have documents in common, for example, in the case of the first and fifth groups; or they might indeed be “false friends”, containing common terms but not common relevant documents, probably the case in the other three groups. In fact, because of the strict definitions of relevance in TREC topics,

- Medicine:
 - postmenopausal estrogen Britain (356)
 - in vitro fertilization (368)
 - anorexia nervosa bulimia (369)
 - health insurance holistic (371)
 - obesity medical treatment (380)
 - alternative medicine (381)
 - mercy killing (393)
- Alternative fuels:
 - hydrogen energy (375)
 - hydrogen fuel automobiles (382)
 - hybrid fuel cars (385)
- Exploited labor:
 - clothing sweatshops (361)
 - human smuggling (362)
- Pharmaceuticals:
 - food/drug laws (370)
 - mental illness drugs (383)
 - orphan drugs (390)
 - R&D drug prices (391)
- Education:
 - mainstreaming (379)
 - teaching disabled children (386)
 - home schooling (394)

Figure 3: A sampling of topics used in the TREC-8 Filtering track, grouped manually into families of related interest.

and that they explicitly seek to limit how far relevance carries to related documents, collaborative filtering techniques might actually harm performance.

4.2 Profile Construction

To build our profiles, we use a technique similar to that used by the AT&T group in TREC-6 [8] and TREC-7 [10]. First, a training collection is constructed from the Financial Times documents from 1992, and all TREC documents from the Foreign Broadcast Information Service, and Los Angeles Times. We gather collection statistics here for all future IDF weights. The training document vectors are weighted log-tfidf, and normalized using the pivoted unique-term document normalization [9].

Pivoted document length normalization is an improvement over the commonly-used cosine normalization. Vector normalization is done in general because longer documents, having more terms, will dominate the similarity calculation otherwise. The cosine normalization does a fairly good job of ensuring that probability of relevance does not increase with length, but still manages to favor long documents. Pivoted normalization repairs this by more “severely” normalizing longer documents.

We then build a routing query using Rocchio’s formula for relevance feedback:

$$Q' = \alpha Q + \beta \left(\frac{1}{|rel|} \sum_{r \in rel} D_r \right) + \gamma \left(\frac{1}{|nrel|} \sum_{n \in nrel} D_n \right)$$

An initial query Q is made from the short topic description, and using it the top 1000 documents are retrieved from the training collection. The results are used to build a feedback query, using:

- Q , the initial short-description query (weighted $\alpha = 3$)
- D_r , all documents known to be relevant to the query in the training collection (weighted $\beta = 2$)
- D_n , retrieved documents 501-1000, assumed to be irrelevant (weighted $\gamma = -2$)

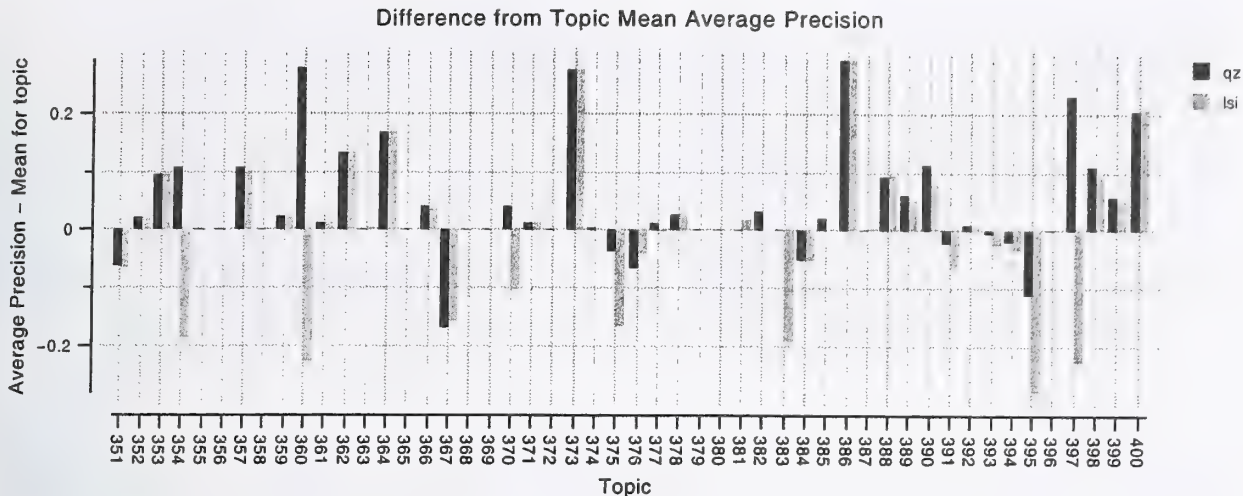


Figure 4: Difference in average precision from mean average precision for each topic. Note that there is very little difference in performance from using LSI.

The set of documents retrieved with the initial query Q is called the “query zone”, and this blind feedback is a kind of unsupervised learning technique. One can also use the top documents from the query zone as unsupervised positive examples, but we found this did not perform as well against the training set. Also, we looked at using the known irrelevant judgments as supervised negatives, but these did not give as good performance on retrieving the training set.

4.3 Results

Our system for routing is based on SMART, with routines added by us for pivoted document length normalization weights, construction of the LSI vector space, and the similarity computations needed to build a ranked list. The LSI code is based on software written at the University of Maryland,² and on SVDPACKC from the NETLIB archive.³ Our experiments were run on a Intel Pentium II-based system running Linux 2.2 with 512MB of RAM and 36 GB of local SCSI-II disk storage.

Two runs were submitted. The first, “umrqz,” used only the routing queries as described above. The second, “umrlsi,” computed an LSI from the collection of these routing queries, and routed the test documents in the resulting LSI space. For LSI to give any benefit, the dimensionality must be reduced below the maximum (in this case, 50 dimensions). We are not aware of any proven principled method for choosing this dimensionality besides trying several levels and seeing what gives the best performance. We thus ran our LSI queries against the training collection at several dimensions, and found that no dimensionality choice seemed to show any benefit for LSI. For the official submission, we arbitrarily chose a 45 dimensions. Overall, both runs performed quite well, with umrqz above the median for 27 queries, and umrlsi for 23. For five queries, we produced the best performance, and for four of those, the LSI gave the maximum score. For the majority of queries, however, there was only a very small difference in performance if any between the two runs. We take this to indicate that good overall performance is mostly due to the routing query construction, which uses a combination of approaches shown to work well in previous TRECs. Figure 4 shows the difference in average precision from the mean score for each topic, illustrating the similarity of the results.

We expected that LSI might not perform much better, because since the topics are mostly different, with little opportunity for overlap, the LSI should have been unable to help most queries. However, for the

²The LSI code is available at <http://www.glue.umd.edu/~oard>

³SVDPACKC is available from <http://www.netlib.org>

example candidate topic “clusters” described above, the difference in average precision from using LSI was negligible. For 18 queries where the difference in average precision between the non-LSI and LSI routing was more than 0.009, in 11 cases the difference was quite small relative to the whole span of scores. In the other seven, the difference was more marked, and in all but one (381) against LSI. For one query (360), LSI gave the minimum performance and the nonrotated query gave the maximum. Furthermore, in the twenty topics where average precision in the umrlsi run was high (> 0.5), precision without LSI was either the same or slightly higher. In eight topics, the LSI average precision was less than 60% of that achieved without LSI. These topics have a fair range of relevant document set sizes and in only one of these topics was performance across all systems poor. One topic in this group was 375, “hydrogen energy”, and three were drug-related (drug legalization, food/drug laws, mental illness drugs). It may be that the drug-related topics contained a lot of shared terms, but this caused LSI to bring out a lot of false friends.

4.4 Discussion

The results indicate that, for the topics and documents here, LSI overall gives no benefit over nontransformed profiles, and if anything may degrade performance among manually-identified clusters of interest. A more in-depth analysis is needed to understand these results. An obvious point for failure might be that the topics have no overlap in relevant documents. If the topics were truly orthogonal, so that there was no overlap among highly-weighted terms among profiles, then we would expect the LSI to give results that are identical to the nontransformed queries, or nearly so. This does seem to match the results as shown in figure 4; however, we know that there are groups of topics which seem to be related. What may in fact be happening here is that collaborating queries are sharing documents which relate to the general interest of the profile, but these documents are not actually *specifically relevant* to the topic as determined by the TREC evaluation.

Alternatively, our profile vectors that we construct may not give a good representation of the topic. Perhaps we are using too many negative example documents, or should be more selective about which terms to retain after the Rocchio expansion. To analyze this, we could look at overlap in terms, training documents, and test documents among the topics. This should give us a better view of where to expect LSI to make gains, but on the other hand this is what the LSI is supposed to do for us. It might be instructive to look at the LSI dimensions and the terms which characterize them, to see what exactly what patterns LSI is finding.

As another possibility, it could be that there are topics which could collaborate, and in fact there is term co-occurrence across their queries which we’d expect the LSI to find, but these patterns aren’t prominent relative to the rest of the collection. This might happen because there aren’t enough terms co-occurring, or the pattern doesn’t span enough queries. In our three example groups, only drug-related topics represent a large segment of the topic collection, and this grouping is vague. An alternative approach might be to augment the matrix used to compute the LSI with more example profiles (perhaps from older TREC topics), or with a sample of documents.

Document Overlap Among Topics. Although Figure 3 implies families of topics that seem to be of related interest, the fact of the matter is that these are separate topics with specific guidelines as to what is and what is not relevant to the topic. Figure 5 gives an illustration of how many topics a document may be relevant to. It shows, for each run and for the relevance judgments, how many (predicted) relevant topics were given for a document. The “qrels” bars show the actual relevance judgments; one can see that the lions share of documents are relevant to only one topic; less than sixty documents are known to be relevant to more than one topic. If a pure collaborative algorithm were used to predict relevance for these topics, and these relevance judgments were sampled for training data, it would fail miserably because the matrix would be too sparse. The probability of any useful quantity of overlap occurring is very small.

The two charts differ in the method for predicting which documents in the umrqz and umrlsi runs are actually relevant. A routing run contains the highest-scored 1000 documents for each topic, but clearly the system does not expect that all 1000 documents are relevant. Thus, we use only predict as relevant some of the documents in each run. The first picks the top 15 ranked documents; 15 is the median number of relevant documents per topic in the actual relevance judgments. The second picks the top 50.

These graphs indicate that our runs tend to spread documents across more topics than are actually relevant. Within the top 15, the qz run distribution is similar to the qrels, and the LSI run gives slightly

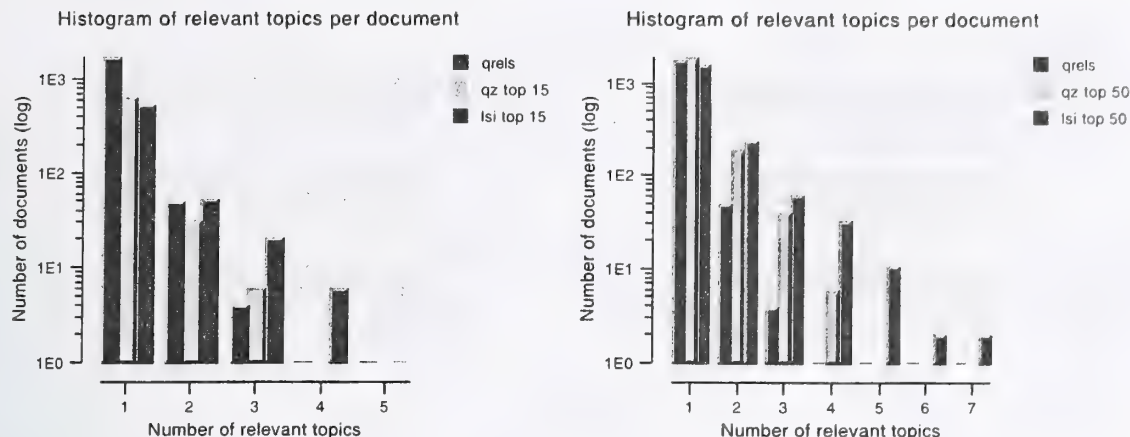


Figure 5: Histograms showing how many topics are relevant to each document, according to TREC-8 relevance judgments, and as predicted by the submitted runs. The horizontal axis is the number of relevant topics; the vertical axis is a log scale of the number of documents which are relevant to only that many topics. The chart on the left uses the top 15 submitted documents in each run; the right uses the top 50.

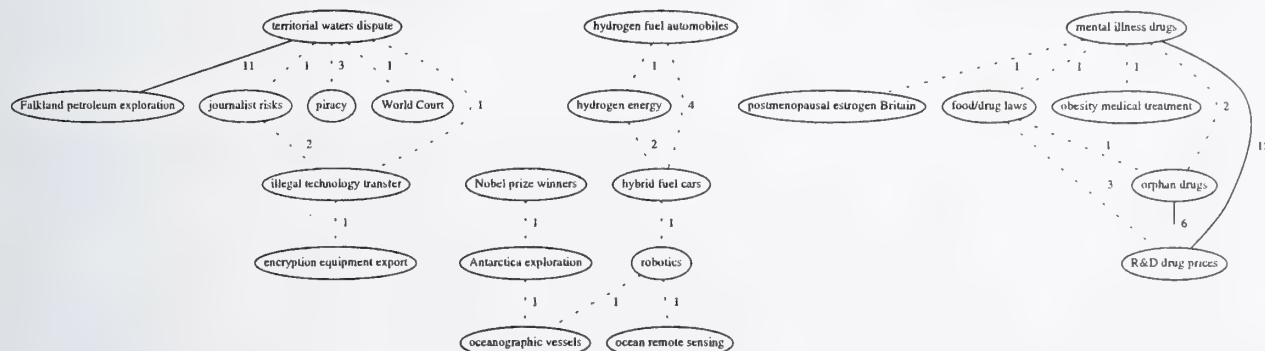


Figure 6: Number of documents shared between topics in the filtering relevance judgments file.

more overlap. At 50 documents per topic the difference is much greater; however, for documents that are shared among only two or three topics, the runs are close to each other in overlap.

Topic Clusters, Revisited. Figure 6 shows how topics share relevant documents, according to the relevance judgments. An edge between two topic nodes indicates a number of documents which are relevant to both topics. The style of line is related to the number of shared documents, as a visual aid; thicker lines indicate more documents. In this diagram, we can see the alternative fuels and pharmaceuticals clusters which we predicted from just reading the topics. These are also loosely linked to other topics, such as “ocean remote sensing”, “robotics” and “obesity medical treatment”. Another strong link exists between “territorial waters dispute” and “Falkland petroleum exploration”, and this group also contains links to “piracy”, “illegal technology transfer”, and “World Court”. Some of these topics are more closely tied together than others, for example, “mental illness drugs” and “R&D drug prices” with 15 documents, while the links between others are more tenuous.

Figure 7 shows the topic relationships as recommended by the query-zone (non-LSI) profiles. The documents represented by these links are in the top 50 for each topic in the submitted run. The graph of the entire recommendation set contains a large number of low-weight links; for clarity, only links of five or more



Figure 7: Document sharing among recommendations made by the query-zoned (non-LSI) profiles. Only links of 5 or more documents are shown.

documents are shown, which in Figure 7 is only 18% of the edge set. We can see that many of the links in the relevance judgments graph are predicted here, although with much larger shared documents sets. We have not manually examined the recommended documents to see what proportion of the shared documents are relevant, and if the stronger links have correspondingly higher numbers of relevant documents.

Although not containing any relevant documents, our education cluster appears, with a link between “teaching disabled children” and “mainstreaming”. We think that the documents along this link may be of related interest but are not actually topically relevant. The query-zone profiles also recommend what we suspect are some red-herring links, for example the links among “cigar smoking”, “health insurance holistic” and “clothing sweatshops”. Another red herring (not strong enough to show on this graph) is a predicted link between “transportation tunnel disasters” and “British Chunnel impact”.

Finally, figure 8 shows document sharing in the top 50 recommendations made by the LSI profiles. Again, this graph only shows links of five or more documents (in this case, 36% of the total edges). The LSI makes some links stronger, bringing them up to our attention when they didn’t appear in the graph for the query zone profiles. One example is the set of topics linked to the Falklands group; most of these links were not strong enough to be visible in figure 7. Another example is in the hybrid fuel cars group; automobile recalls wasn’t linked heavily before, but it is now and is a red herring. Also, note that the pharmaceuticals and medicine topics are more closely linked in the LSI recommendations.

The LSI also is lessening the impact of some relationships in the feedback profiles. The links between “hybrid fuel cars” and “hydrogen fuel automobiles” is slightly stronger while the links to both of these from “hydrogen energy” is slightly weaker. This effect is not as strong as we would have hoped, though.

In summary, it seems that the collaborative LSI technique serves to enhance ties among clusters of topics, but that these ties consist of related but essentially irrelevant documents, as can be seen by comparing the graphs of the runs to the relevance judgments graph. It is also likely that our naive term selection in the profile expansion step, especially with respect to negative examples, is calling forth more red herrings than we might find otherwise.

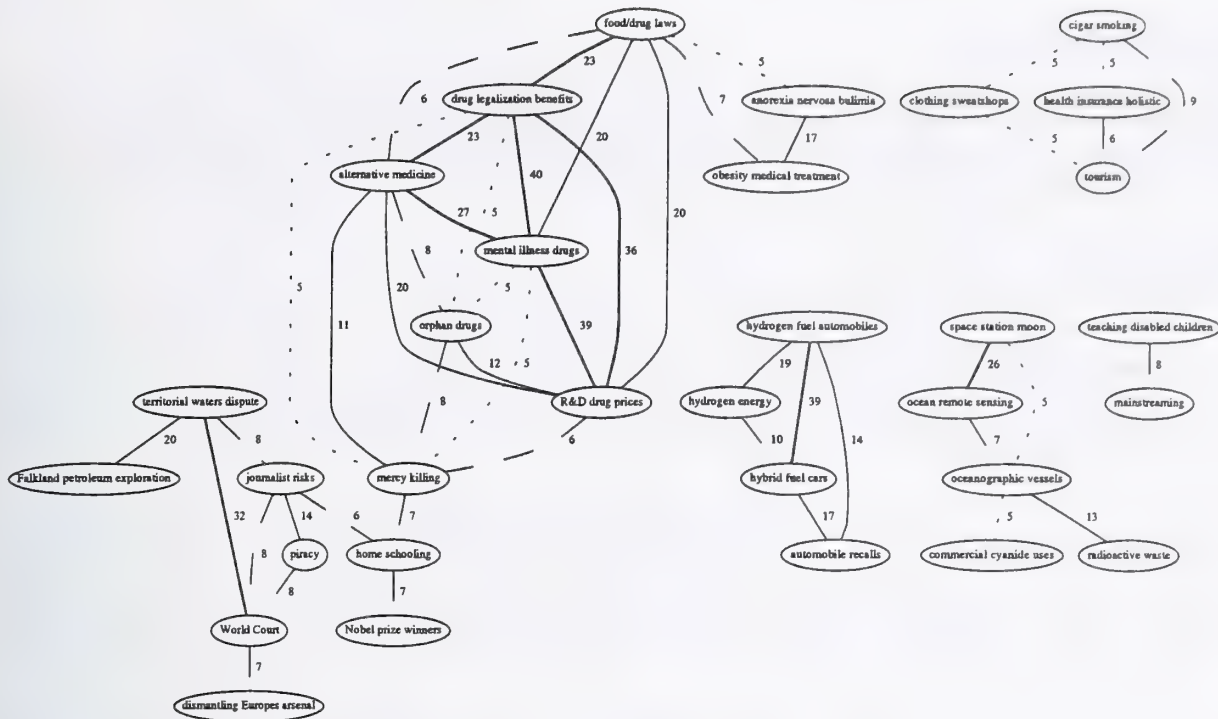


Figure 8: Document sharing among recommendations made by the LSI profiles. Only links of 5 documents or more are shown.

5 Ad Hoc Task

For TREC-7 and TDT-2 we had been using PRISE, but our interest in trying out Pirkola's technique for CLIR led to our choice of Inquiry for CLIR TREC-8. The Ad Hoc task provides a useful opportunity for us to get new people familiar with the tools that we will be using in the CLIR track—this year we submitted a single official Ad Hoc run using Inquiry 3.1p1 with the default settings. Queries were automatically formed from the title and description fields, and we automatically performed limited stop structure removal based on a list of typical stop structure observed in earlier TREC queries (e.g., “A relevant document will contain”).

6 Conclusion

Our investment in TREC this year was rewarded with a rich set of insights. In Cross-Language Information Retrieval, we learned that we can construct passable systems using freely available resources but that a more efficient implementation of Pirkola's method may be needed before interactive applications will be practical. In this first year of the Question Answering track, we learned that the techniques we have been working with have good potential and that the evaluation methodology is quite tractable. In the Routing task, we achieved competitive results using a new approach, and recognized some promising directions for future work. The great frustration of TREC is that there are so many important and well framed questions being explored, but that practical considerations make it necessary for each team to focus on only a few. We have explored the potential for building a larger team through both on-campus and off-campus collaborations this year, and have been quite pleased with the result. Perhaps the strongest legacy of this effort, then, will be the closer personal and professional ties that we have forged.

Acknowledgments

CLIR work was supported in part by DARPA contract N6600197C8540 and a Shared University Research equipment grant from IBM. QA work was supported in part by Communication Security Establishment under contract W2213-9-0001/001/SV awarded to Nalante, Inc. and by Natural Sciences and Engineering Research Council of Canada grant OGP121338. Routing work was supported by the Department of Defense.

References

- [1] Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, December 1995.
- [2] Susan T. Dumais. Using LSI for information filtering: TREC-3 experiments. NIST Special Publication 500-225, pages 219–230, Gaithersburg, MD, November 1995.
- [3] Fredric C. Gey, Hailing Jiang, Aitao Chen, and Ray R. Larson. Manual queries and machine translation in cross-language retrieval and interactive retrieval with Cheshire II at TREC-7. In *The Seventh Text REtrieval Conference*, pages 527–540, November 1998. <http://trec.nist.gov>.
- [4] David Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference (SIGIR '94)*, pages 282–291, Dublin, Ireland, July 1994.
- [5] Gina-Anne Levow and Douglas W. Oard. Evaluating lexicon coverage for cross-language information retrieval. In *Workshop on Multilingual Information Processing and Asian Language Processing*, pages 69–74, November 1999.
- [6] Douglas W. Oard. TREC-7 experiments at the University of Maryland. In *The Seventh Text REtrieval Conference (TREC-7)*, pages 541–545, November 1998. <http://trec.nist.gov>.
- [7] Ari Pirkola. The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–63, August 1998.
- [8] Amit Singhal. AT&T at TREC-6. In E. M. Voorhees and D. K. Harman, editors, *The Sixth Text REtrieval Conference*, NIST Special Publication 500-240, pages 215–226, Gaithersburg, MD, November 1997.
- [9] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, August 1996.
- [10] Amit Singhal, John Choi, Donald Hindle, David D. Lewis, and Fernando Pereira. AT&T at TREC-7. In E. M. Voorhees and D. K. Harman, editors, *The Seventh Text REtrieval Conference*, NIST Special Publication 500-242, pages 239–252, Gaithersburg, MD, November 1998.
- [11] Ian M. Soboroff and Charles K. Nicholas. Combining content and collaboration in text filtering. In Thorsten Joachims, editor, *Proceedings of the IJCAI'99 Workshop on Machine Learning in Information Filtering*, pages 86–91, Stockholm, Sweden, August 1999.

INQUERY and TREC-8

James Allan[†], Jamie Callan*, Fang-Fang Feng[†], and Daniella Malin[†]

[†]Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts USA

*Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania USA

This year the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts participated in seven of the tracks: ad-hoc, filtering, spoken document retrieval, small web, large web, question and answer, and the query tracks. We spent significant time working on the filtering track, resulting in substantial performance improvement over TREC-7. For all of the other tracks, we used essentially the same system as used in previous years.

In the next section, we describe some of the basic processing that was applied across most of the tracks. We then describe the details for each of the tracks and in some cases present some modest analysis of the effectiveness of our results.

1 Tools and Techniques

Although UMass used a wide range of tools, from Unix shell scripts, to PC spreadsheets, four major tools and techniques were applied across almost all tracks: the Inquiry search engine, the InRoute filtering engine, query processing, and a query expansion technique known as LCA. This section provides a brief overview of each of those so that the discussion does not have to be repeated for each track.

1.1 Inquiry

All tracks other than the filtering track used Inquiry[Callan et al., 1992] as the search engine, sometimes for training, and always for generating the final ranked lists for the test. We used Inquiry V3.2, an in-house development version of the Inquiry system made available by the CIIR (V3.1). The differences between the two are not consequential for this study.

The current belief function used by Inquiry to calculate the belief in term t within document d is:

$$w_{t,d} = 0.4 + 0.6 \times \frac{tf_{t,d}}{tf_{t,d} + 0.5 + 1.5 \frac{\text{length}(d)}{\text{avg len}}} \times \frac{\log \frac{N+0.5}{n_t}}{\log N + 1}$$

where n_t is the number of documents containing term t , N is the number of documents in the collection, “avg len” is the average length (in words) of documents in the collection, $\text{length}(d)$ is the length (in words) of document d , and $tf_{t,d}$ is the number of times term t occurs in document d .

*The work reported was carried out while Jamie Callan was at the University of Massachusetts.

1.2 InRoute

The InRoute filtering system is based on the same Bayesian inference network model as InQuery. It is designed to operate efficiently in high-volume filtering environments, where incoming documents must be processed rapidly, one at a time. It uses the same document indexing techniques, query language, and scoring algorithms as InQuery. Corpus statistics for the document stream are estimated using an archival corpus or are learned as documents stream by. InRoute also incrementally learns improved profiles [Allan, 1996] and improved thresholds [Callan, 1998], as relevance judgments become available for documents that have been disseminated.

InRoute was used only in the filtering track.

1.3 Query Processing

The processing of queries—i.e., the transformation from TREC topic to InQuery query—was handled very similarly to the way it was managed for TREC-7, though there were some small changes. It includes three steps:

1. Basic Query Processing removes stop words and phrases (e.g., “relevant documents will include”), and stop structures—i.e., that are sentences discussing criteria of non-relevance in the narratives. For removing the stop structures the processor simply segments each sentence first then removes the sentences that contain the stop structure (e.g., “Documents discussing ... are not relevant”), but keep those clause which stands on the negative part of the removed sentence (such as in “... not relevant, unless ...”, where the “unless” clause is not be removed).
2. Query Formalizing identifies noun phrases (as in TREC-6 and 7), and proper names. The proper names were transformed to use the ordered proximity-one operator (e.g., `#passage25(#1(Golden Triangle))`, which requires that the two words occur immediately adjacent in the text in that order; the passage operator affects the weighting of the feature. For noun phrases we not only used the phrase operator but also duplicated the single terms used by the phrase (e.g., `#passage25(#phrase(tropical storms))`, `tropical`, `storms`). Note that for proper names, the single term duplication was *not* done.

The query formalizing also identifies compound words, like *wildlife* and *airport*, that are formalized with the synonym operator (e.g., `#syn(#1(air port) airport)`). While the query is formalized, if there is any word concerned with foreign countries, like *international*, *world*, or *Europe*, a token `#foreigncountry` will be added to the query. If there is a term concern with the United States, then a token `#usa` will be added. If both `#foreigncountry` and `#usa` are found in a query, all such tokens are removed.

Finally a query is formed with the weighted sum operator (`#wsum`) with a weight for each term. For those terms occurring in the title and description fields the weight 1.0 is used, while 0.3 is used for those terms occurring in the narrative field. That is, we trust the title and the description more than the narrative.

3. Query Expansion adds 50 LCA concepts to each query. These 50 concepts are collected from top 30 passages (the passage database are built with TREC volumes 1 through 5). This is the same process that we used in TREC-7, but we did not use “filter-required” on the title words this year. LCA is described next.

1.4 Local Context Analysis (LCA)

In SIGIR '96, the CIIR presented a query expansion technique that worked more reliably than previous “pseudo relevance feedback” methods.[Xu and Croft, 1996] That technique, Local Context Analysis (LCA),

locates expansion terms in top-ranked passages, uses phrases as well as terms for expansion features, and weights the features in a way intended to boost the expected value of features that regularly occur near the query terms.

LCA has several parameters that affect its results. The first is the choice of LCA database: the collection from which the top ranked passages are extracted. This database could be the test collection itself, but is often another (perhaps larger) collection that it is hoped will broaden the set of likely expansion terms. In the discussion below, if the LCA database is not the test collection itself, we identify what collection was used.

LCA's other two parameters are the number of top passages used for expansion, and the number of expansion features added to the query. The LCA features were put into a query construct that allows a weighted average of the features. Assuming n features, f_1 through f_n , they are combined as:

$$\begin{array}{rcl} \#wsum(& 1.0 & f_1 \\ & \vdots & \vdots \\ & 1 - (i - 1) * 0.9/s & f_i \\ & \vdots & \vdots \\ & 1 - (n - 1)0.9/s & f_n) \end{array}$$

Here, s is scaling factor that is usually equal to n . The weighted average of expansion features is combined with the original query as follows:

$$\#wsum(1.0 \ 1.0 \text{ original-query} \quad w_{lca} \text{ lca-wsum})$$

where w_{lca} is the weight that the LCA features are given compared to the original query. Note that the final query is a weighted combination of the original query and the expansion features. As will be discussed below, in the SDR track the combination was unintentionally done differently, slightly shifting the balance between the original query and the expansion concepts.

2 Ad-hoc Track

Other than minor changes to the query processing, we did not investigate any ideas in the ad-hoc track this year. We submitted four runs, each of which used the complete set of query processing techniques outlined in Section 1.3. The runs and their average precision were:

INQ601	title only	0.2325
INQ602	description only	0.2492
INQ603	title plus description	0.2659
INQ604	title, description, and narrative	0.2809

INQ603 (comparatively, our best run) is below the average for 13 topics of the 50 topics. We have not analyzed those runs in any additional detail.

3 Filtering Track

Our goals for the Filtering track were to test the InRoute filtering system and a new threshold learning algorithm. We felt that performance in TREC-7 was quite poor for three reasons:

1. disseminating too many non-relevant documents early on;
2. not monitoring query performance during the run; and,
3. using a threshold learning method that could only raise thresholds.

We account for each of these in TREC-8 resulting in substantial performance improvement. We dealt with them as follows:

- The first problem was that too many documents were disseminated at an early stage. Wishing to mimic closely a real world situation where we cannot know an appropriate threshold initially, we tried to learn thresholds quickly from the first retrieved documents. This proved too difficult. We were not able to find a reasonably effective threshold without retrieving (and taking the hit for) a sizeable number of documents, most of which would turn out to be non-relevant. As a solution, we ran each query on a retrospective database and chose an initial threshold on the basis of these alternative document scores. This turned out to be highly effective. This technique was adopted from CLARIT's TREC-7 submission.[Zhai et al., 1998]
- Our second problem was that we had not implemented any sort of performance monitoring mechanism to watch queries during the course of the run and "shut off" queries for which performance was far below the target precision. This year, we simply stopped retrieving documents for queries retrieving more than N non-relevant documents and MIN or fewer relevant documents. MIN was somewhat arbitrarily set to 2. Any less than that would not have been reasonable given the threshold learning method we chose (described below). Because of the effectiveness of our initial thresholds, we were able to set N to a fairly small number; N was also the parameter we varied between submissions. We set $N = 9$ and $N = 15$ for our official submission for LF1 and LF2 respectively. These values were chosen on the basis of test runs using AP data and TREC-7 queries. A more elegant solution would have been to choose N and MIN based on some function of the target precision as embodied in utility metric being used for evaluation. Unfortunately, we did not have time before the submission deadline to discover what this function should be.
- In our previous threshold learning method, thresholds could only go up and never down during the course of a run. This meant that the threshold learning was insensitive to new trends in scores. It also increased the importance of the initial phases of learning. We needed to retrieve many documents at the beginning of the run in order to get a good start. This year we opted for a threshold learning method that could adjust thresholds both up and down. In doing so, we gave up both modifying the queries on the basis of retrieved documents and dynamically adjusting term idfs.

Our method this year was to set thresholds to the lowest document score, in a list of scores sorted highest to lowest, at which precision was equal to or greater than the target precision. We collected document scores and after each new arrival, sorted the scores and computed precision at each point. We then chose the first precision point to equal or exceed the target precision. Since this process would be futile if none of the documents retrieved were relevant, we used our initial thresholds until at least MIN (two) documents had been retrieved. Document scores of initial documents would have very little in common with document scores of subsequent documents if the queries and/or idfs had changed during the course of the run. In the interest of threshold sensitivity, we chose not to modify our queries or adjust idfs. Our indexing process did not make use of the controlled-language field in the FT database.

3.1 Queries for filtering

We used the title, description and narrative parts of the topic processed the queries using all three of the query processing steps described for our ad-hoc runs. However, `#usa` and `#foreigncountry` tokens were never used, regardless of whether the query suggested they should be.

	FT92	FT93	FT94	FT92-94
INQ610 LF1	35	39	42	32
INQ611 LF2	36	30	33	25
INQ612 LF2	38	36	38	32
INQ613 LF1	34	31	34	28

Table 1: Number of queries at or above the median for filtering

	FT92	FT93	FT94	FT92-94
INQ610 LF1	258	29	30	350
INQ611 LF2	88	-14	24	74
INQ612 LF2	125	29	9	139
INQ613 LF1	127	-186	-129	-155

Table 2: Sum of our scores minus sum of median scores for filtering

3.2 Filtering Results

The CIIR submitted four runs, two for evaluation using the LF1 utility metric and two for evaluation using the LF2 utility metric. As discussed above, we varied the number N , of non-relevant retrieved documents we allowed the system to retrieve before shutting off the query. We allowed that number to be greater in our primary run for LF2 than LF1 since LF2 is the more lenient of the two metrics. In the end, the smaller value for N outperformed the larger value for both metrics. The runs are as follows:

INQ610	$N = 9$	LF1 primary run
INQ611	$N = 15$	LF2 primary run
INQ612	$N = 9$	LF2 secondary run
INQ613	$N = 15$	LF1 secondary run

Tables 1 and 2 show the performance of this run compared to the median.

4 Spoken Document Retrieval Track

We applied the full query processing technique and ran the resulting queries. Note that unlike in past years, we did the LCA query expansion on an external collection rather than on the SDR collection itself. We participated only in the “quasi-SDR” version of the track.

5 Small Web Track

For the two gigabyte Web track, our results were reasonable. Of the 50 topics, 42 were better than the median and only 7 worse. Three of our scores were the maximum score for that query. The run, INQ620 used the same query set as Ad-hoc INQ603, i.e, the title and description portions of the topic, with the same query processing. We did not take advantage of link structure in any way. The average precision of the run was 0.3327 with precision at 10 documents of 0.5040 and at 20 of 0.4130.

6 Large Web Track

We submitted one run to the Large Web Track. That run, INQ650, achieved a modified average precision of 0.3927. Its precision at 10 documents retrieved was 0.4960 and at twenty documents, it was 0.5000.

7 Question and Answer Track

Our work in the Q&A track is in loose combination with Sheffield University. We used straightforward IR techniques to isolate passages of text that were highly likely to contain the answer to the questions. We submitted those passages as our “answers” and also handed them to Sheffield to do some further processing. At this point, we do not have comparative information to show the impact of their work.

We used Inquiry’s best passage operator to try to locate the answer for a particular question in the 5 top retrieved documents. Inquiry’s best passage operator is flexible with respect to the size of the best passage the user is looking for. This setting is in word count rather than bytes. We set the best passage size to 10 words for the 50-byte runs and 50 words for the 250-byte passage runs. We then post-processed the answer to pull out something of the correct size. If a word was split by the 50- or 250-byte boundary, we did not include the partial word in our “answer.”

CIIR submitted four runs, two for the 50-byte limit and two for the 250-byte limit. The difference between these runs was whether or not we used LCA expansion on the questions (i.e., whether step 3 of the query processing in Section 1.3 was applied). We found that LCA expansion improved performance. Overall we did quite well:

Runid	> med	≥ med	< med	Our average	Average of median	% Difference
INQ634 - 50 bytes, with expansion	43	185	20	0.18737	0.11970	+57.1%
INQ635 - 250 bytes, with expansion	58	172	33	0.37828	0.28081	+34.7%
INQ638 - 50 bytes, no expansion	29	176	29	0.12556	0.12485*	+0.5%
INQ639 - 250 bytes, no expansion	45	165	40	0.33283	0.28081	+18.5%

(*)The average of the median is different between these two 50 byte runs apparently because NIST carried the second evaluation out to two rather than just one significant digit.)

8 Query track

The CIIR contributed a large set of queries to the Query track. They were generated as part of a class assignment for a database class in the Fall of 1998.

9 Acknowledgements

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623. It was also supported in part by Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235. Additional support came in part from Air Force Research

Laboratory Contract number F30602-98-C-0110. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

- [Allan, 1996] Allan, J. (1996). Incremental relevance feedback. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*, pages 270–278, Zurich. Association for Computing Machinery.
- [Callan, 1998] Callan, J. (1998). Learning while filtering documents. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval*, Melbourne. Association for Computing Machinery.
- [Callan et al., 1992] Callan, J. P., Croft, W. B., and Harding, S. M. (1992). The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, Valencia, Spain. Springer-Verlag.
- [Xu and Croft, 1996] Xu, J. and Croft, W. B. (1996). Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*, pages 4–11, Zurich. Association for Computing Machinery.
- [Zhai et al., 1998] Zhai, C., Jansen, P., Stoica, E., Grot, N., and Evans, D. (1998). Notes on threshold calibration in adaptive filtering: The CLARIT system TREC-7 filtering report. In *The Seventh Text REtrieval Conference TREC-7*.

IRIS at TREC-8

Kiduk Yang, and Kelly Maglaughlin
School of Information and Library Science
University of North Carolina
Chapel Hill, NC 27599-3360 USA
{yangk, maglk}@ils.unc.edu

0 Submitted Runs

unc8al32, *unc8al42*, *unc8al52* – Category A, automatic ad-hoc task runs
unc8iap – interactive track run

1 Introduction

We tested two relevance feedback models, an adaptive linear model and a probabilistic model, using massive feedback query expansion in TREC-5 (Sumner & Shaw, 1997), experimented with a three-valued scale of relevance and reduced feedback query expansion in TREC-6 (Sumner, Yang, Akers & Shaw, 1998), and examined the effectiveness of relevance feedback using a subcollection and the effect of system features in an interactive retrieval system called IRIS (Information Retrieval Interactive System¹) in TREC-7 (Yang, Maglaughlin, Mehol & Sumner, 1999).

In TREC-8, we continued our exploration of relevance feedback approaches. Based on the result of our TREC-7 interactive experiment, which suggested relevance feedback using user-selected passages to be an effective alternative to conventional document feedback, our TREC-8 interactive experiment compared a passage feedback system and a document feedback system that were identical in all aspects except for the feedback mechanism. For the TREC-8 ad-hoc task, we merged results of pseudo-relevance feedback to subcollections as in TREC-7.

Our results were consistent with that of TREC-7. The results of passage feedback, whose system log showed high level of searcher intervention, was superior to the document feedback results. As in TREC-7, our ad-hoc results showed high precision in top few documents, but performed poorly overall compared to results using the collection as a whole.

2 IRIS: Key Components

IRIS is an interactive retrieval system designed to provide users with ample opportunities to interact with the system throughout the search process. For example, users can supplement the initial query with two-word collocations suggested by the system, perform relevance feedback by selecting relevant documents or passages, or add and delete query terms. IRIS, first created in 1996 at the School of Information and Library Science at UNC-CH, has been under continuous development, evolving with each participation in TREC experiments. Below is a description of its key components used in TREC-8 experiments.

2.1 Text Processing

IRIS processes the text first by removing punctuation, and then excluding the 390 high-frequency terms listed in the WAIS default stopwords list as well as “IRIS stopwords,”² which were arrived at by examining the inverted index and identifying low frequency terms that appeared meaningless.

¹ IRIS was first developed by Kiduk Yang, Kristin Chaffin, Sean Semone, and Lisa Wilcox at the School of Information and Library Science (SILS) at the University of North Carolina. They worked under the supervision of William Shaw and Robert Losee.

² IRIS stopwords are defined as all numeric words, words that start with a special character, words consisting of more than 25 non-special characters, and words with embedded special characters other than a period, apostrophe, hyphen, underline, or forward or backward slash.

After the punctuation and stopword removal, IRIS conflates each word by applying one of the four stemmers implemented in the IRIS Nice Stemmer module,³ which consists of a simple plural remover (Frakes & Baeza-Yates, 1992, chap. 8), the Porter stemmer (Porter, 1980), the modified Krovetz inflectional stemmer (Krovetz, 1993),⁴ and the Combo stemmer that uses the shortest whole word (i.e., word that appears in a dictionary) returned by the three stemmers. We used the Krovetz stemmer in TREC-7 for its conservative conflation tendencies, but we opted for the simple plural remover in TREC-8 to speed up the indexing time. Simple stemmer was chosen over the porter stemmer to minimize the overstemming effect, with the hope that understemming effect would be compensated by the feedback query expansion process.

2.2 Phrase Construction

Phrase construction method employed in our TREC-7 interactive experiments was used to construct the phrase indexes in TREC-8. Using the online dictionary and the clause recognition algorithm built into the Nice Stemmer, we constructed a two-word noun-noun phrase index by first extracting adjacent word pairs of noun and proper noun combinations within a clause,⁵ and then discarding the phrases occurring 20 or less times in the collection to reduce indexing time and to conserve computer resources. The phrase occurrence frequency threshold of 20 was arrived at by selecting the number that produced the phrase index whose size was most comparable to that of the collocation index. To augment the proper nouns in the online dictionary, all capitalized words not occurring at the beginning of a sentence were considered to be proper nouns. Hyphenated words were broken up and stemmed by the simple plural remover before the noun-noun phrase construction module was applied. Hyphenated words in their raw form (i.e. as they appear in documents sans punctuation) were added to the index as well.

2.3 Ranking Function and Term Weights

IRIS ranks the retrieved documents in decreasing order of the inner product of document and query vectors,

$$\mathbf{q}^T \mathbf{d}_i = \sum_{k=1}^t q_k d_{ik}, \quad (1)$$

where q_k is the weight of term k in the query, d_{ik} is the weight of term k in document i , and t is the number of terms in the index. We used SMART *Lnu* weights for document terms (Buckley, Singhal, Mitra, & Salton, 1996; Buckley, Singhal, & Mitra, 1997), and SMART *lrc* weights (Buckley, C., Salton, G., Allan, J., & Singhal, A., 1995) for query terms. *Lnu* weights attempt to match the probability of retrieval given a document length with the probability of relevance given that length (Singhal, Buckley, & Mitra, 1996). Our implementation of *Lnu* weights was the same as that of Buckley et al. (1996, 1997) except for the value of the *slope* in the formula, which is an adjustable parameter whose optimal value may depend, in part, on the properties of the document collection.

According to the pre-test experiments, an *Lnu* slope of 0.5 performed best with feedback, especially when using both single term and phrase indexes. Based on these findings, we used a slope of 0.5 to optimize performance with feedback.

2.4 Feedback Models

2.4.1 Adaptive Linear Model

We used the same implementation of the adaptive linear model (Wong & Yao, 1990; Wong, Yao, Salton, & Buckley, 1991) in TREC-8 as in TREC-7. The basic approach of the adaptive linear model, which is based on

³ Nice stemmer was implemented by Kiduk Yang, Danqi Song, Woo-Seob Jeong, and Rong Tang at SILS at UNC. For an interactive demonstration, please visit <http://ils.unc.edu/iris/nstem.htm>.

⁴ The modified Krovetz inflectional stemmer implements a modified version of Krovetz's inflectional stemmer algorithm and restores the root form of plural ("-s," "-es," "-ies"), past tense ("-ed"), and present participle ("-ing") words, provided this root form is in our online dictionary.

⁵ IRIS identifies a clause boundary by the presence of appropriate punctuation marks such as a comma, period, semicolon, question mark, or exclamation mark.

the concept of the preference relation from decision theory (Fishburn, 1970), is to find a *solution vector* that, given any two documents in the collection, will rank a more-preferred document before a less-preferred one (Wong et al., 1988).

In the relevance feedback interface of IRIS, users can evaluate documents as “relevant,” “marginally relevant,” or “nonrelevant.” By adapting the concept of the user preference relation to extend the relevance scale from a binary to a three-valued scale, we constructed the following formula for the starting vector. Note that this formula can be adjusted for any multivalued relevance scale:

$$\mathbf{q}_{(0)} = c_0 \mathbf{q}_{rk} + \frac{c_1}{N_{new\ rel}} \sum_{new\ rel} \mathbf{d} + \frac{c_2}{N_{new\ mrel}} \sum_{new\ mrel} \mathbf{d} - \frac{c_3}{N_{new\ nonrel}} \sum_{new\ nonrel} \mathbf{d}, \quad (2)$$

where \mathbf{q}_{rk} is the query vector that produced the current ranking of documents; c_0 , c_1 , c_2 , and c_3 are constants; $N_{new\ rel}$, $N_{new\ mrel}$, and $N_{new\ nonrel}$ are the number of new relevant, new marginally relevant, and new nonrelevant documents respectively in the current iteration; and the summations are over the appropriate new documents. The detailed description of the adaptive linear model can be found in Yang et. al. (1999).

2.4.2 Passage Feedback Model

The conventional relevance feedback models assume the user’s relevance judgement to be about an entire document and treat documents as the information units. The unit of a document, however, is sometimes determined by arbitrary reasons such as convenience or convention rather than content, which can produce a document containing subsections of various information content. In such instances, the user’s determination of relevance is likely to be based on certain portions of a document rather than the entirety of it. Even in a document of consistent information content, the user may be interested in only a specific information described in certain passages of the document. Thus, we think that the conventional practice of using document as the unit of feedback may be less effective than using user-defined passages in relevance feedback.

To test this theory, we implemented the “passage feedback model” in TREC-7 with the following formula for feedback vector creation:

$$\mathbf{q}_{new} = \mathbf{q}_{old} + \sum_{rel} \mathbf{p} - \sum_{nonrel} \mathbf{p}, \quad (4)$$

where \mathbf{q} is the query vector and \mathbf{p} is the passage vector determined by the user’s selection of the relevant and nonrelevant portions of documents. Since the normalization factor of the Lnu weight is based on document length, an inverse document frequency weight was used for the passage vector \mathbf{p} . The passage feedback approach differs fundamentally from the philosophy of the adaptive linear model in that it simply expands the query vector to make it more “similar” to relevant passages and “dissimilar” to nonrelevant passages rather than trying to rank a document collection in the preference order defined by a training set

Though the underlying implementations of the passage feedback model are the same in our TREC-7 and TREC-8 interactive experiments, we made a significant modification to the user interface in TREC-8. One of the prevalent user comments of our TREC-7 passage feedback system was about the clunkiness of its feedback interface. Users had to first select passages by highlighting with mouse, copy the highlighted portions, toggle to the passage feedback window, and then paste the copied selection into appropriate windows. System logs as well as user comments seemed to indicate the difficulty of these steps required for the passage feedback, which we thought kept users from fully enjoying the benefits of the passage feedback system. Consequently, we simplified the feedback interface by using an embedded java applet, which consolidated document display window and feedback window as well as simplified the overall passage feedback operation.

3 Pre-test Experiments

In our TREC-8 pre-test experiments, we compared two feedback query expansion size (300 terms, 30 terms) as well as several pseudo-feedback approaches using TREC-7 queries and relevance judgements on the full TREC-7 document collection. In keeping with the experimental design of both TREC-7 and TREC-8, queries were first submitted to each subcollection of FBIS, Federal Register, Financial Times, and LA Times, after which several pseudo-feedback runs were executed, and finally the results were merged by the similarity scores to produce the top 1000 ranked documents.

The pseudo-feedback runs consisted of using the full length queries with 100th document as non-relevant and various number of top documents as relevant with additional parameter of 300-term feedback vector vs. 30-term feedback vector. The effect of changing the number of pseudo-relevant documents was negligible, but 300-term feedback vector outperformed 30-term vector, which is consistent with finding from previous experiments.

4 Ad-hoc Experiments

4.1 Research Question

We continued exploration of our approach of subcollection retrieval in TREC-8 ad-hoc experiments. Though we are aware that retrieval performance of using the whole collection is superior to that of using subcollections with initial retrieval, we wanted to see if we could minimize the performance loss with relevance feedback. If the subcollection retrieval with simple pseudo-relevance feedback can be shown to be competitive to that of using a whole collection, then subcollection retrieval may be a desirable strategy in real world situations, where the whole document collection statistics is unavailable or too costly to compute.

In this light, We posed the following question.

- Is the subcollection retrieval results using pseudo-relevance feedback competitive to the initial retrieval results using the whole collection?

4.2 Research Design

There are two main issues in the subcollection retrieval as we have defined it. First, there is the problem of “collection fusion”, where various methods of merging the results of subcollection retrieval have been examined (Dumais, 1993; Voorhees, Gupta, & Johnson-Laird, 1995; Savoy, Calve, & Vrajitoru, 1997). Then there is the question of how best to implement the pseudo-feedback process, which has been one of the main concerns of TREC ad-hoc participants in the past.

Our research design in TREC-8 was strongly influenced by the desire to lay the groundwork for building a large scale system such as IRISWeb⁶. Consequently, our approach was to use the simpler methods over more complex ones in order to increase the system efficiency. We selected the top two performing subcollection retrieval approaches from the pre-test experiments as well as a run with a medium length query and a shorter feedback vector. The final results were produced by first retrieving 10% of documents in each collection and merging the results by their raw query-document similarity scores:

- *unc8al32*: Pseudo-relevance feedback with the top 5 retrieved documents as relevant and the 100th document as non-relevant using the top 250 positive-weighted terms and the lowest 50 negative-weighted terms.
- *unc8al42*: Pseudo-relevance feedback with the top 10 retrieved documents as relevant and the 100th document as non-relevant using the top 250 positive-weighted terms and the lowest 50 negative-weighted terms.
- *unc8al52*: Pseudo-relevance feedback with the top 3 retrieved documents as relevant and the 100th document as non-relevant using the top 25 positive-weighted terms and the lowest 5 negative-weighted terms.

⁶ IRISWeb is an experimental Web search engine, which is a variation of IRIS system. Please see <http://ils.unc.edu/iris> for more information.

Both *unc8al32* and *unc8al42* used the full query (i.e. title, description, narrative), whereas *unc8al52* used only the title and description fields to construct the queries. The default system constructs for the ad-hoc experiment were:

- *Lnu* 0.5 weights for documents, and *ltc* weights for queries
- adaptive linear model for feedback
- use of single-term and noun-noun phrase index
- conflation by removal of simple plurals

4.3 Results

The TREC-8 ad-hoc collection consists of 130,471 FBIS, 55,630 Federal Register, 210,158 Financial Times, and 131,896 LA Times documents. Each document collection was first processed individually to generate single-word indexes of 244,458 terms and phrase index of 60,822 terms for FBIS, 118,178 single and 28,669 phrases terms for Federal Register, 290,880 single and 87,144 phrases terms for Financial Times, and 228,507 single and 62,995 phrase terms for LA Times collection.

TREC evaluation measures of the top 1000 documents (Table 2) showed consistent results with pre-test, where the variation in the number of relevant documents of the pseudo-relevance feedback made little difference (*unc8al32* vs. *unc8al42*). The best performance of the three was achieved by using the medium length query and shorter feedback query (*unc8al52*), which was somewhat unexpected. To investigate this matter further, we ran a series of post analysis runs, where we tested the effect of the initial query length in combination with feedback vector length. The results consistently showed the runs with medium length query to perform better than ones with full length query regardless of the feedback vector length. It appears that better initial retrieval result achieved using the medium length query (Table 1) overpowers any advantage gained by the longer feedback vector.

It is unclear why our system performed better with the medium length query, whereas the usage of full length query has shown to be advantageous by the best performing ad-hoc systems in the past (Voorhees & Harman, 1999). It is possible that we need better query processing and expansion approaches to minimize the noise in the narrative portion of the query vector while maximizing its descriptiveness.

As expected, our performance using subcollection retrieval is somewhat worse off than the median performance of TREC-8 participants (Table 3). Table 1 and 2 show the subcollection retrieval results to be competitive to retrieval results using the whole collection. The gap in performance levels between subcollection and whole collection retrieval is narrowed by feedback, which suggests the possibility that simple pseudo-relevance feedback methods may be more effective in a subcollection retrieval setting.

Table 1. Initial Retrieval Results of top 1000 documents

	Subcollection Retrieval		Whole Collection Retrieval	
	long query	medium query	full query (<i>unc8alb1</i>)	medium query (<i>unc8alb2</i>)
Average Precision	0.1031	0.1406	0.1687	0.1715
Precision at 5 docs	0.2200	0.3480	0.4320	0.4040
Precision at 10 docs	0.2080	0.2980	0.3820	0.3600
Precision at 100 docs	0.1460	0.1612	0.1824	0.1686
Number of Relevant Documents	2272	2272	2262	2287

Table 2. Pseudo-feedback Results of top 1000 documents

	<i>unc8al32</i>	<i>unc8al42</i>	<i>unc8al52</i>	<i>unc8alb1*</i>	<i>unc8alb2**</i>
Average Precision	0.1347	0.1372	0.1669	0.1722	0.1746
Precision at 5 docs	0.3280	0.3280	0.4120	0.4160	0.4040
Precision at 10 docs	0.3080	0.3160	0.3580	0.3800	0.3700
Precision at 100 docs	0.1642	0.1664	0.1690	0.1846	0.1738
Number of Relevant Documents	2249	2261	2281	2285	2290

* Same as *unc8al42* except for using the whole collection

** Same as *unc8al52* except for using the whole collection

Table 3. Best, Median, Worst Results (top 1000 documents) of all TREC8 ad-hoc participants

	<i>long query*</i>			<i>medium query**</i>		
	<i>Best</i>	<i>Median</i>	<i>Worst</i>	<i>Best</i>	<i>Median</i>	<i>Worst</i>
Average Precision	0.4338	0.2570	0.0186	0.4339	0.2261	0.0001
Number of Relevant Documents	3854	2784	598	3807	2791	37

* Statistics computed over 37 automatic ad hoc runs that used the entire topic statement.

** Statistics computed over 59 automatic ad hoc runs that used the title and description sections of the topic statement (4 runs used description only).

5. Interactive Experiment

5.1 Research Question

In our TREC-7 interactive experiments, we examined the effects of user interface on retrieval performance by comparing a system with complex interface with one with a simpler interface. We also compared in TREC-7 the effectiveness of a “passage feedback” system, where user-defined passages were used to expand the feedback query, with a conventional “document feedback” system that used relevance judgement based on documents to perform the relevance feedback using the adaptive linear model.

In keeping with our hypothesis, passage feedback system results were better than that of the document feedback results in our TREC-7 experiments. However, the results of the simple interface system versus the complex interface system was rather unexpected in that it performed slightly worse than the complex interface system. After further examination, which revealed more searcher intervention steps in the system logs of the complex system, we concluded that complex system allowed users more opportunities to intervene, thereby positively affecting the retrieval performance.

Furthermore, we noticed that the passage feedback features in TREC-7 were somewhat underutilized. Though there were more retrieval iterations per query in the passage feedback system than in the document feedback system, there were more reformulation of the initial query cycles than the expansion of the feedback vector by using the passage feedback interface. One of the prevalent user comments of our TREC-7 passage feedback system was about the clunkiness of its feedback interface. Indeed, our TREC-7 passage feedback interface required several keystrokes or mouse clicks including the toggling between two windows.

Based on these observations, we hypothesized the following in our TREC-8 experiment:

- Passage feedback in an interactive system can perform better than the document feedback.
- Improving the usability of the passage feedback interface will invite more usage of it, thereby resulting in more positive user intervention.
- User intervention can positively affect the retrieval performance.

5.2 Methodology

To test our hypothesis, we constructed a passage feedback system with a streamlined feedback interface for our TREC-8 interactive experiment and compared its performance with that of a document feedback system. If our hypothesis were correct, our TREC-8 passage feedback system should show better results and more user

intervention steps than both our document feedback system in TREC-8 and our “difficult” passage feedback system in TREC-7.

The passage feedback system and the document feedback system in TREC-8 are identical in all aspects except for how relevance feedback is implemented. Both systems have exactly the same features and interfaces for initial query formulation (Figure 1), initial query modification (Figure 2), and feedback query modification (Figure 4). The one and only difference between systems occurs in the relevance feedback interface. The document feedback system employ the conventional feedback mechanism of judging the relevance of a document as a whole (Figure 3.1) using the adaptive linear model, but the passage feedback system allows users to select relevant and nonrelevant portions of a document with which to expand the feedback query vector (Figures 3.2).

User intervention can occur in the initial query modification phase where the user can supplement the initial query with “suggested phrases” selected by the system, in the feedback query modification phase where the user can add new terms or delete existing terms from the feedback query, and in the relevance feedback phase.

The underlying system constructs for both systems were essentially the same as that of the ad-hoc system except for interactive feedback mechanism. Document term weights of *Lnu* 0.5 and *lrc* query term weights were used to maximize the relevance feedback influence, while the feedback query with 250 terms with highest positive weights and 50 terms with lowest negative weights was used in order to optimize the system for efficiency. A phrase index of adjacent noun-noun pairs were also used in suggesting potentially useful phrases for the initial query as well as in expanding the feedback vectors.

5.3 Searchers

Table 4 shows the information about each searcher's background and search experience gathered by pre-study questionnaires. All searchers were either working on or had received a graduate degree, most (17) of which were in Library and Information Science. The searchers had been searching between 1 and 14 years, with 4.5 being the average. Nine of the 24 searchers were male.

Table 4. Response Frequency of Searchers on Pre-Study Questionnaire

	<i>No Experience</i>		<i>Some Experience</i>		<i>Great Deal of Experience</i>
1.Using a point-and-click interface			3	5	18
2.searching elec. library catalogs		2	3	9	10
3.searching on CD ROM systems	1	3	12	8	
4.searching commercial systems	5	11	2	5	1
5.using WWW search services			5	8	11
6.searching other systems	4		2	1	2
	<i>Never</i>	<i>Once or twice a year</i>	<i>Once or twice a month</i>	<i>Once or twice a week</i>	<i>Once or twice a day</i>
7.Searching frequency			2	9	13
	<i>Strongly Disagree</i>	<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>	<i>Strongly Agree</i>
8.Enjoys information searches		1	6	10	7

5.4 Results

The performance of IRIS measured by the mean instance precision (MIP) and mean instance recall (MIR) measures confounded both our hypothesis and the previous results of the passage feedback system. The results of the TREC-8 passage feedback system with an “improved” interface was the worst among all our systems tested in both TREC-7 and TREC-8, which is the exact opposite of what we expected (Table 5). However, the difference between systems ($H_a: \mu_{df} > \mu_{pf}$) was not statistically significant in either MIP ($p=0.34$) or MIR ($p=0.09$).

Table 5. Interactive Experiment Result Statistics

	<i>TREC-8</i>		<i>TREC-7</i>	
	<i>document feedback (df)</i>	<i>passage feedback (pf)</i>	<i>document feedback</i>	<i>passage feedback</i>
Mean Instance Precision	0.643	0.625	0.673	0.778
Mean Instance Recall	0.277	0.228	0.281	0.314

A further examination of the results with system order consideration suggests that the passage feedback might have been more difficult to learn than the document feedback system (Table 6). In comparing the results of the first system shown to the searchers, the document feedback system outperforms the passage feedback system. This is not true when they were the second systems searched. Also, the passage feedback system's precision and recall scores improve when the system was the second system searched. There was actually a slight decrease in precision and recall in the document feedback system when it was the second system searched.

Table 6. Order Effects in TREC-8 Systems

	<i>First System Searched</i>		<i>Second System Searched</i>	
	<i>document feedback (df1)</i>	<i>passage feedback (pf1)</i>	<i>document feedback (df2)</i>	<i>passage feedback (pf2)</i>
Mean Instance Precision	0.685	0.575	0.600	0.676
Mean Instance Recall	0.295	0.196	0.259	0.260

Table 7 shows the p-values of various system differences. The system difference with statistical significance ($\alpha=0.05$) occurs between the first systems searched, thus giving evidence to the hypothesis that the searchers would do better with the document feedback system as the first system. The improvement in performance of the passage feedback system ($H_a: \mu_{pf1} > \mu_{df1}$) is statistically significant in MIP ($\alpha=0.05$), and not significant in MIR. Overall lack of significance in system learning effect suggests that either there was not sufficient time for any system learning to take place, or that there is no system learning to be gained in the first place.

Table 7. Statistical Significance (p-values) of System Differences

	$H_a: \mu_{df1} > \mu_{pf1}$	$H_a: \mu_{pf2} > \mu_{df2}$	$H_a: \mu_{df1} > \mu_{df2}$	$H_a: \mu_{pf2} > \mu_{pf1}$
Mean Instance Precision	0.03	0.11	0.08	0.05
Mean Instance Recall	0.02	0.49	0.26	0.09

The examination of the system logs show less user intervention in the passage feedback system than the document feedback system, as was the case in our TREC-7 interactive experiments. Even with the improved interface, the searchers seemed to have more difficulty using the passage feedback system. Given the time constraint of the experiment, the searchers might be more comfortable making document-level relevance judgement by quickly scanning documents than having to identify relevant passages, which would possibly require more time and mental effort.

In conclusion, we believe the poor results of the passage feedback system is largely due to our failure to make the passage feedback system more usable. Though we improved the passage selection interface, we did little to ease the cognitive burden of having to identify relevant passages rather than documents. Thus, the main challenge for the passage feedback system lies in helping searchers identify relevant passages quickly and easily.

References

- Buckley, C., Salton, G., Allan, J., & Singhal, A. (1995). Automatic query expansion using SMART: TREC 3. In D. K. Harman (Ed.), *Overview of the Third Text REtrieval Conference (TREC-3)* (NIST Spec. Publ. 500-225, pp. 69-80). Washington, DC: U.S. Government Printing Office.
- Buckley, C., Singhal, A., & Mitra, M. (1997). Using query zoning and correlation within SMART: TREC 5. In E. M. Voorhees & D. K. Harman (Eds.), *The Fifth Text REtrieval Conference (TREC-5)*.
- Buckley, C., Singhal, A., Mitra, M., & Salton, G. (1996). New retrieval approaches using SMART: TREC 4. In D. K. Harman (Ed.), *The Fourth Text REtrieval Conference (TREC-4)* (NIST Spec. Publ. 500-236, pp. 25-48). Washington, DC: U.S. Government Printing Office.
- Dumais, S. T. (1993). LSI meets TREC. In D. K. Harman (Ed.), *Proceedings of the First Text REtrieval Conference (TREC-1)*, 137-152.
- Fishburn, P. C. (1970). *Utility theory for decision making*. New York: John Wiley & Sons.
- Frakes, W. B., & Baeza-Yates, R. (Eds.). (1992). *Information retrieval: Data structures & algorithms*. Englewood Cliffs, NJ: Prentice Hall.
- Krovetz, R. (1993). Viewing morphology as an inference process. *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 191-203.
- Porter, M. (1980). An algorithm for suffix stripping. *Program*, 14, 130-137.
- Savoy, J., Calve, A., & Vrajitoru, D. (1997). Report on the TREC-5 experiment: Data fusion and collection fusion. In E. M. Voorhees & D. K. Harman (Eds.), *The Fifth Text REtrieval Conference (TREC-5)*.
- Singhal, A., Buckley, C., & Mitra, M. (1996). Pivoted document length normalization. *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 21-29.
- Sumner, R. G., Jr., & Shaw, W. M., Jr. (1997). An investigation of relevance feedback using adaptive linear and probabilistic models. In E. M. Voorhees & D. K. Harman (Eds.), *The Fifth Text REtrieval Conference (TREC-5)*.
- Sumner, R. G., Jr., Yang, K., Akers, R., & Shaw, W. M., Jr. (1998). Interactive retrieval using IRIS: TREC-6 experiments. In E. M. Voorhees & D. K. Harman (Eds.), *The Sixth Text REtrieval Conference (TREC-6)*.
- Voorhees, E., Gupta, N. K., & Johnson-Laird, B. (1995). The Collection fusion problem. In E. M. Voorhees & D. K. Harman (Eds.), *Overview of the Third Text REtrieval Conference (TREC-3)*.
- Voorhees, E., & Harman, D. K.. (1999). Overview of the Seventh Text REtrieval Conference (TREC-7). In E. M. Voorhees & D. K. Harman (Eds.), *The Seventh Text REtrieval Conference (TREC-7)*.
- Wong, S. K. M., & Yao, Y. Y. (1990). Query formulation in linear retrieval models. *Journal of the American Society for Information Science*, 41, 334-341.
- Wong, S. K. M., Yao, Y. Y., Salton, G., & Buckley, C. (1991). Evaluation of an adaptive linear model. *Journal of the American Society for Information Science*, 42, 723-730.
- Yang, K., Maglaughlin, K., Meho, L., & Sumner, R. G., Jr. (1999). IRIS at TREC-7. In E. M. Voorhees & D. K. Harman (Eds.), *The Seventh Text REtrieval Conference (TREC-7)*.

Figure 1 Initial Query Formulation Interface

IRIS Initial Search: Financial Times - Netscape

IRIS Initial Search: Financial Times

IRIS userID: Topic Number

Enter your query (e.g. What is the meaning of life?)

To emphasize a word, put + at the end e.g. word+
 To indicate a phrase without component words as search terms, use single hyphen: e.g. word1-word2
 To indicate a phrase with component words as search terms, use double hyphen: e.g. word1--word2

IRIS Home Page | Search Other TREC Database | IRIS Help index | Comments & Questions

Figure 2 Initial Query Modification Interface

IRIS - Netscape

Initial Query Modification Page

[View Initial Query](#) | [Help for this page](#) | [New Search](#)

Terms Entered			Phrases Suggested		
ADD	term	#documents	ADD	phrases	#documents
<input checked="" type="checkbox"/>	country	40961	<input checked="" type="checkbox"/>	import-sugar	40
<input checked="" type="checkbox"/>	cuban	353	<input type="checkbox"/>	country-import	93
<input checked="" type="checkbox"/>	import	7895	<input type="checkbox"/>	country-status	21
<input checked="" type="checkbox"/>	sugar	1774	<input type="checkbox"/>	east-import	21
			<input type="checkbox"/>	country-entry	21
			<input type="checkbox"/>	country-event	21
			<input type="checkbox"/>	single-country	21
			<input type="checkbox"/>	country-shop	21
			<input type="checkbox"/>	country-note	21
			<input type="checkbox"/>	country-strength	21
			<input type="checkbox"/>	fuel-import	21
			<input type="checkbox"/>	country-favour	21
			<input type="checkbox"/>	import-history	21

Initial Query: topic #000

What countries import Cuban sugar?

Figure 3.1 Document Feedback Interface

Relevance Feedback Iteration #1
View Initial Query | Help for this page

Relevant?
SAVE Yes Maybe No

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	FT93: Commodities and Agriculture: Russia seen importing less
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	FT92: Commodities and Agriculture: Bolivia allows sugar imports
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	FT92: Commodities and Agriculture: Pakistan can halt sugar imports
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	FT92: Commodities and Agriculture: Pakistani government plans to
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	FT92: Commodities and Agriculture: Pakistan can halt sugar imports

DOCUMENT #38068

YR92 / Commodities and Agriculture: Pakistan can halt sugar imports

By FARHAN BOKHARI

Story From: ISLAMABAD

SURPLUS SUGAR stocks in Pakistan are set to eliminate the country's need to import sugar, although the potential for export remains unclear. Pakistan's total production is expected to hit 2.6m tonnes by the end of the 1992-93 fiscal year, up from 2.3m tonnes in 1991-92. Government officials estimate that private traders are holding 132,000 tonnes of stocks, and that is expected to start rising next month as the new production season begins at sugar mills.

The rise in sugar production has partly resulted from an increase in the number of sugar mills as well as improvements in the recovery rate of sugar cane and beet. Pakistan started with 2 sugar mills with a daily sugar cane crushing capacity of 1,500 tonnes at the time of its independence in 1947; today there are 54, with an aggregate capacity of 175,000 tonnes.

The rise in production has allowed cuts in sugar imports. In June this year, imports of white refined sugar fell to just 538 tonnes, down from 3,480 tonnes in May. Last year, 36,819 tonnes was imported in June,

Figure 3.2 Passage Feedback Interface

Relevance Feedback Iteration #1
View Initial Query | Help for this page

Relevant?
SAVE

<input type="checkbox"/>	FT93: Commodities and Agriculture: Russia seen importing less
<input checked="" type="checkbox"/>	FT92: Commodities and Agriculture: Bolivia allows sugar imports
<input checked="" type="checkbox"/>	FT92: Commodities and Agriculture: Pakistan can halt sugar imports
<input type="checkbox"/>	FT92: Commodities and Agriculture: Pakistani government plans to
<input type="checkbox"/>	FT93: Commodities and Agriculture: Russia to tax sugar imports
<input type="checkbox"/>	FT94: Russia cuts off Cuba's oil supplies

DOCUMENT #38068

YR92 / Commodities and Agriculture: Pakistan can halt sugar imports

By FARHAN BOKHARI

Story From: ISLAMABAD

1992-93 fiscal year, up from 2.3m tonnes in 1991-92. Government officials estimate that private traders are holding 132,000 tonnes of stocks, and that is expected to start rising next month as the new production season begins at sugar mills.

The rise in sugar production has partly resulted from an increase in the number of sugar mills as well as improvements in the recovery rate of sugar cane and beet. Pakistan started with 2 sugar mills with a daily sugar cane crushing capacity of 1,500 tonnes at the time of its independence in 1947; today there are 54, with an aggregate capacity of 175,000 tonnes.

The rise in production has allowed cuts in sugar imports. In June this year, imports of white refined sugar fell to just 538 tonnes, down from 3,480 tonnes in May. Last year, 36,819 tonnes was imported in June, following 48,290 tonnes in May.

However, the country's sugar export potential remains unclear. With countries such as Brazil and Cuba having lower costs of production, Pakistani sugar might not be able to compete, said one senior official. Up to 100,000 tonnes of sugar is estimated to be smuggled annually to neighbouring Iran and Afghanistan. That has made it difficult to assess if a surplus will be left after meeting domestic consumption, including smuggling, in order to set aside large quantities for export. However, Pakistan will at least save valuable foreign exchange by meeting its sugar

of production, Pakistani sugar might not be able to compete.

Figure 4 Feedback Query Modification Interface

Feedback Query #1 Modification Screen

[Add New Terms](#) | [Help for this page](#) | [Saved Documents](#) | [New Search](#)

[Continue Search](#) | [Back to Default Selections](#)

Top 30 Distinguishing Terms of Feedback Query			
Positive Terms		Negative Terms	
import-sugar	<input checked="" type="checkbox"/>	raw	<input checked="" type="checkbox"/>
cuban	<input checked="" type="checkbox"/>	raw-sugar	<input checked="" type="checkbox"/>
sugar	<input checked="" type="checkbox"/>	centre	<input checked="" type="checkbox"/>
import	<input checked="" type="checkbox"/>	figure	<input checked="" type="checkbox"/>
country	<input checked="" type="checkbox"/>	study	<input checked="" type="checkbox"/>
dollar	<input checked="" type="checkbox"/>	moscow	<input checked="" type="checkbox"/>
export	<input checked="" type="checkbox"/>	russia	<input checked="" type="checkbox"/>
pakistan	<input checked="" type="checkbox"/>	reuter	<input checked="" type="checkbox"/>
set	<input checked="" type="checkbox"/>	reuter	<input checked="" type="checkbox"/>
domestic	<input checked="" type="checkbox"/>	company	<input checked="" type="checkbox"/>
mill	<input checked="" type="checkbox"/>	fall	<input checked="" type="checkbox"/>
mill-sugar	<input checked="" type="checkbox"/>	group	<input checked="" type="checkbox"/>
production	<input checked="" type="checkbox"/>	unchanged	<input checked="" type="checkbox"/>

Initial Query: topic #000

What countries import Cuban sugar?

Displayed terms on the left are added to the feedback query by default. Please deselect the terms you wish to exclude from the feedback query.

Moving More Quickly toward Full Term Relations in Information Space

Gregory B. Newby*

School of Information and Library Science
University of North Carolina at Chapel Hill

Abstract

This paper describes the ISpace retrieval system's involvement in TREC8. The main goal for this year's work was to speed up document indexing and query processing compared to previous years. This goal was achieved, but retrieval performance was not as good as for TREC7. System details for the AdHoc task, small Web task, and large Web (VLC) task are presented. The AdHoc task emphasized query expansion, while the large Web track emphasized rapid indexing and retrieval. The paper describes an implementation of a multidimensional tree structure for retrieval from information space based on the kd-tree. The larger setting for ISpace, the TeraScale Retrieval project, is summarized. A concluding section describes plans for ISpace.

Introduction

Efforts for the 8th Text REtrieval Conference (TREC) included the following:

1. AdHoc task, fully automatic.
2. Small Web track
3. Large Web track (VLC)

Throughout the work described here, the central question of interest is:

How might information space techniques achieve high performance?

The issue of performance is ambiguous, but was defined as emphasizing the following, in decreasing order of importance:

- a. Performance means being able to quickly produce a ranked response set for a query topic
- b. Performance means being able to handle the full variety of queries and documents – i.e., without limitations on the number of unique terms or number of documents
- c. Performance means the response set has a large proportion of relevant documents

In this hierarchy, the goal of high relevance is uncharacteristically last, but not forgotten. Because post-hoc analysis of last year's non-judged TREC submissions (Newby, 1999) indicated reasonable recall-precision performance with exact precision of 0.14, the emphasis was on developing a more practical and usable system. While 0.14 is unremarkable compared to other groups' TREC submissions, it represented an order of magnitude improvement from prior years (Newby, 1998).

A description of the information space technique, system design considerations for each phase of the work and outcomes follow. A concluding section summarizes this year's TREC activities and lays out plans for the near future.

* School of Information and Library Science, Campus box 3360 Manning Hall, Chapel Hill, NC, 27599-3360.
Email: gbnewby@ils.unc.edu

The Larger Picture

ISpace, the Information Space retrieval system described here, is part of a larger research project. The project, led by the author, is called TeraScale Retrieval. The purpose of the TeraScale Retrieval project is to investigate problems of information retrieval through the development and evaluation of high-performance modular retrieval system components.

In Korfhage et al. (1999), an effort was made by leaders in retrieval research to identify major challenges to progress. At about the same time, the U.S. Presidential Information Technology Advisory Committee (PITAC, 1999) released a report intended to chart the near-term future of high-technology research and research funding. Both of these reports had similar thing to say about the needs of retrieval research. The TeraScale Retrieval project is attempting to meet some of these needs:

1. "To develop specifications for [and implement] a complete and modular set of IR tools to be made available to the IR community (Korfhage et al., p. 5)."
2. To create an infrastructure for "sharing and distribution of IR tools (Korfhage et al., p. 5).
3. To overcome limitations of many retrieval research systems, viz., "they lack modularity and do not facilitate interactive and operational retrieval experimentation (Korfhage et al., p. 5). This comment was directed particularly at the limitations of TREC.
4. To develop software "for managing large amounts of information (PITAC, p. 4)."

PITAC recognized that "transforming the way we deal with information ... requires significant improvements in data access methods, including high performance information systems and tools to help individuals locate information and present, integrate, and transform the information in meaningful ways (1999, p. 13)."

ISpace is part of a hybrid system. It uses data specifications and structures similar to that of IRIS (Yang & Maglaughlin, 2000). It is able to perform Boolean retrieval, and integrates multiple methods for basic tasks such as stemming and file access. It is being expanded to model vector space, probabilistic, and latent semantic indexing styles of retrieval. This will provide a controlled environment for retrieval experimentation that will enable IR researchers to control differences of implementation details among the various major types of retrieval systems.

One of the most important tasks of the TeraScale Retrieval project is to address scaling issues. Future information retrieval systems will be aimed at terabytes of raw data: millions of unique terms, billions of documents, and peta-scale quantities (quadrillions) of sub-documents. In spite of the continuation of Moore's law for doubling CPU power and processing speed every 18 months, the quantity of information we would wish to access is growing more quickly.

Many IR methods, including the most common implementations of vector spaces, probabilistic and Boolean models, use search and indexing algorithms that scale approximately linearly with the size of the collection. With modern hardware, these produce acceptable performance with collections the size of TREC AdHoc (2GB). But consider: if a search that takes 1 second on a 2GB collection takes 100 seconds on a 200GB collection, we can't wait for Moore's law to bring this longer search time back to 1 second (two orders of magnitude). Instead, we need to develop methods that scale better than linearly to speed up performance. The multidimensional tree described for the Large Web task below is one such method.

The TeraScale Retrieval project is also intended to address the shortcomings of large Web search engine's contributions to the scientific knowledge of information retrieval. Although these giants have made significant contributions to people's ability to find information on the Web, they generally have not shared their particular methods with the scientific community. This is consistent with PITAC's finding, "the PITAC members from industry were unanimous in their opinion that it is not feasible for the private sector to assume responsibility for long-term, high-risk research, in spite of the success of the information technology industry (p. 6)." The TeraScale Retrieval project will specifically address large-scale Web-based retrieval issues, and share findings, software and systems with the community of scientists interested in information retrieval.

A Brief Tour of Information Space

This section introduces the approach to information retrieval employed by the author. The Information Space approach to information retrieval is comparable to Latent Semantic Indexing (LSI, see Deerwester et al., 1990), with some differences. These differences are:

- Information Space starts with the term by term correlation matrix while LSI starts with the term by document co-occurrence matrix;
- Information Space performs eigensystems analysis while LSI performs a Singular Value Decomposition (SVD);
- Information Space has not made use of eigenvalues for scaling document vectors while LSI does make use of singular values (the square root of the eigenvalues);
- The Information Space approach does not assume that higher-dimensioned eigenvectors are without merit while LSI historically has sought to discard higher-dimensioned eigenvectors

Techniques that have been applied to various types of IR have also been applied to the Information Space system described here, called ISpace. These include: query expansion, part of speech tagging, document length normalization, term weighting, and stemming.

Like many other IR systems, ISpace may be compared to the Vector Space Model (VSM). Although variations in the VSM have proliferated, fundamental differences between Information Space and the VSM are:

- Information Space measures relations among terms while VSM treats term vectors are unrelated (orthogonal)
- A fair approximation of an Information Space may be visualized in 2 or 3 dimensions while a vector space does not have a clear visual interpretation

For TREC8, the ISpace programs from prior years were largely rewritten. The goal of rewriting was to increase modularity and enable incorporation of multiple IR techniques. For example, the same data structures, term weights, etc. used for an Information Space retrieval experiment with ISpace may also be used for a VSM or Boolean retrieval experiment.

The specific steps taken for ISpace retrieval are described for the different tasks, below.

The AdHoc Task

Experimentation with query expansion was the main goal for the AdHoc task. The steps taken for the task were as follows:

1. Read terms from all AdHoc documents, building an inverted index and term frequency list. About 367,000 Porter-stemmed terms were pre-identified from various word lists, past TREC topics, and the AdHoc document set. A test run demonstrated that all possible terms could be indexed, but would include many single-use terms not likely to be in queries.
2. 3136 non-stoplist terms were selected based on IDF values for inclusion in the information space. (The SMART stoplist was used.)
3. A term co-occurrence matrix for the 3136 terms was generated from the inverted index.
4. A Pearson product moment correlation matrix of term relations was generated from the co-occurrence matrix.
5. Eigensystems analysis was performed on the correlation matrix, resulting in 3062 eigenvalues ranging from 2871 to nearly zero.
6. Each AdHoc document was then re-analyzed and assigned a location vector at the geometric center of the eigenvectors of the terms it contained.
7. These document vectors were weighted with $tf * idf$ using simple formulas from Frakes & Baeza-Yates (1992). Then, vectors were normalized to unit length.
8. Topics were expanded by either 25 or 50 terms. Term expansion added the most highly correlated terms with each topic term.
9. The topic vector in the information space was then weighted and normalized at the center of its (expanded) term vectors.
10. The closest document vectors to the topic vector were retrieved in rank order

Four AdHoc runs were submitted. Two judged runs utilized topic titles and descriptions, expanded by 25 or 50 terms (isa25, isa50). Two un-judged runs utilized titles only and expanded by 25 or 50 terms (isa25t, isa50t). Results were considerably poorer than comparable AdHoc ISpace runs from TREC7, yielding average exact precision scores under 0.025.

There are two likely explanations for this poor showing. One is that part of speech tagging (Brill, 1994) and query processing, used last year, are important. This is supported by the observation that expanded topics expanded all topic terms, including terms without much discriminatory value. This may have served to bring in useful terms, but it also increased the noise in topics. For query processing, all topics had been pre-analyzed for sentence structure in TREC7 so that phrases about non-relevance were eliminated. Thus, only "desirable" terms were kept.

The other likely explanation is that the 3136 terms chosen were not well suited to the TREC8 AdHoc topics. Of the 283 unique stemmed terms in topics 401-450, 14 topic terms that casual interpretation suggests were important were not among them.

The stemmed missing terms were: Burma, comet, decai, estonia, hurrican, legionnair, lockerbi, milosev, parkinson, potassium, saharan, salvag, scotland, and typhoon. From this list, it is evident that low performance would be expected from the topics that include them: 403, 405, 406, 408, 409, 411, 415, 423, 429, 434 and 443.

However, topics 403 ("osteoporosis") and 408 ("tropical storms") both performed relatively well, as mentioned below. It appears that the missing term from 403, "potassium," was not

overly injurious, and the presence of both “tropical” and “storm” may have offset the missing “hurricane” and “typhoon” from topic 408.

Exact precision scores for the AdHoc task ranged from 0 to .59. Table 1 presents a summary of scores. Due to a scoring error, scores for the title-only 50-word expansion are not considered here.

Topics with exact precision or average precision greater than 0.10 on the AdHoc task included:

1. Topic 403, “osteoporosis.” This topic had the highest median average precision of all topics, possibly due to a large number of fairly specific terms in the expanded topic. Even the title-only run yielded good scores (Exact precision = .42). Term expansion appeared to work well, with a better exact precision for expansion by 50 than by 25 (.38 vs. .19).
2. Topic 407, “poaching, wildlife preserves.” Good performance on the title-only run, but not title plus description. .13 average precision and .17 exact precision for title-only.
3. Topic 408, “tropical storms.” Average precision scores of about .9 on all runs were unexceptional, but exact precision scores of .22 and above for all but the expand by 25 condition indicate reasonable early precision.
4. Topic 441, “Lyme disease.” This seems to be a good example of a topic well-represented by two terms. Title-only runs yielded exact precision scores in excess of .52, and average precision of over .58, while title plus description runs were very poor, below .01 on both exact and average precision.
5. Topic 444, “supercritical fluids.” This is the one topic where title plus description heavily out-scored title-only, but only for the expansion by 50 case. 3 of the 17 (total) relevant documents across participants’ systems were retrieved in the top 10 documents, indicating some good terms were brought in by expansion that were missed in other runs.

Table 1: Summary of AdHoc Task Retrieval Performance

	Expand by 25 (isa25)	Expand by 50 (isa50)	Expand by 25 title only (isa25t)
Exact precision	0.0081	0.0349	0.0515
Average precision	0.0026	0.0203	0.0273
Number over median average precision	0	0	0

The easiest interpretation to make from these results is that query term expansion is valuable, but only when applied to useful terms. Future efforts will return to a reliance on part of speech tagging and term weights to identify “good” terms to expand (i.e., terms with good discrimination potential), while avoiding expanding the rest.

Finally, note that no runs were made with non-expanded queries. This would be a valuable comparison. Future experiments should therefore compare:

- Expansion versus no expansion
- Different levels of expansion (bringing in 25 terms versus 50 terms, or other values)
- Methods for choosing terms to expand (*tf* weights, part of speech, presence in <title> field, etc.)

The Small Web Track

The steps taken for the small Web track (a 2 gigabyte subset of the large Web track's Very Large Corpus or VLC) were nearly identical to the steps for AdHoc. The only difference was that the co-occurrence matrix was not based on the small Web track corpus, but rather the AdHoc corpus. This mismatch should theoretically be detrimental to the recall-precision statistics, but with an average exact precision of 0.02 from the AdHoc task, such differences would be difficult to see. (Performance at exact precision values of 0.02 or below is approximately equal to that expected from a random sample of documents, although higher early precision than later precision indicates the retrieved set is actually non-random.).

Four small Web track runs were submitted. As with the AdHoc task, two title plus description runs were judged, but two title only runs were not judged. Due to a scoring error, scores for the title-only 50-word expansion are not considered here. See Table 2 for a summary.

Table 2: Summary of Small Web Track Retrieval Performance

	Expand by 25 (isw25)	Expand by 50 (isw50)	Expand by 25 title only (isw25t)
Exact precision	0.0048	0.0451	0.0524
Average precision	0.0018	0.0291	0.0291
Number over median average precision	0	2	2

Topics yielding exact precision or average precision over 0.10 on one or more runs included:

1. Topic 403, "osteoporosis." Presumably for the same reasons the topic performed well for the AdHoc task (above).
2. Topic 408, "tropical storms." Interestingly, performance was somewhat less than for the AdHoc task, but still better than most other topics.
3. Topic 415, "drugs, Golden Triangle." This topic performed well with title-only, but not well with title plus description, presumably due to the ambiguous terms in the description (e.g., "organizations," "international.").
4. Topic 432, "profiling, motorists, police." This topic also performed well only on title-only runs.
5. Topic 433, "Greek, philosophy, stoicism." One or more good expansion terms were presumably identified in title plus description expansion by 50, which yielded .18 exact precision but less than .01 for the other runs.
6. Topic 448, "ship losses." Similarly to topic 433, exact precision on title plus description expansion by 50 yielded an exact precision of .18, but very low scores on other runs.
7. Topics 441, 445, 446 and 450 all had comparable patterns of exact precision scores near or over .20, but unexciting average precision scores (except for topic 450, with an average precision of .10 or above for all runs). These topics are characterized by useful title terms and, except for 441, specific description terms.

The small Web track data set seemed to be fairly well suited for topics 401-450. Average precision scores across all TREC participants were sometimes higher for the Web track, and

other times higher for the AdHoc task. The mean of median average precision scores for the AdHoc task and small Web track across all TREC8 participants was identical to 2 decimal places at .24.

It seems reasonable to conclude that the somewhat better performance for the small Web track versus the AdHoc task for the ISpace system is genuine. Some possible explanations are that an increased variety in Web documents resulted in fewer highly-ranked false hits, and that the 3136 terms pre-identified for information space document location calculation helped to eliminate a larger proportion of small Web track documents than AdHoc task documents.

The Large Web Track (VLC)

The large Web track was the main emphasis of this year's work, with the primary goal of achieving reasonable speed and efficiency in processing documents and topics. As discussed below, this goal was achieved, but at the expense of retrieval performance. This section describes the system considerations for the large Web track, including a discussion of the tree data structure employed for retrieval. However, because results for the VLC were submitted after the deadline, they were not officially judged. *Because almost none of the VLC documents retrieved by ISpace were judged, no useful retrieval performance statistics were generated.*

As with the small Web track, the large Web track made use of the information space pre-computed from the AdHoc task. Thus, the steps for retrieval were nearly identical, except that a multidimensional tree was used for retrieval, rather than a brute-force search for the closest document vectors for each query vector.

Although 3062 eigenvectors (dimensions) were available for use from the AdHoc collection, the VLC documents were only processed with 300 eigenvectors per term. The 300-dimensional information saved to disk was not utilized for the actual retrieval, however: only 100 dimensions were utilized. This is beneath the threshold recommended by Deerwester et al. (1990) and others, but was necessary to fit as many documents as possible into the tree, which was memory-based.

Table 3: Summary of Large Web Track Non-Retrieval Performance

Measure	Performance	Notes
Time to index 100GB	5 clock hours	0.0012 seconds per document
Index size, all files	11GB, 18 files	~640 bytes per document
Index size, eigenvectors only	8GB, 3 files	300 dimensions per document
Time to start retrieval engine (prior to running queries)	20 clock minutes	
Time to run 10,000 queries	52 seconds	0.0052 seconds per query

The 100GB VLC (Very Large Corpus) was stored on a disk/tape array at UNC Chapel Hill. This enabled rapid staging from tape robot to disk without requiring more than about 10GB of disk space at a time to stage documents. See Table 3 for a summary of non-retrieval performance measures

Most of the large Web track was performed on UNC-CH's Sun ES-10000 server with 36 processors and 16GB of main memory. Disk access on the disk/tape array was found to be quite fast, cutting indexing time by 60% versus a comparable server with local SCSI RAID disk. No

parallelization was utilized (all programs ran on a single CPU), and the ES-10000 was shared with many other user processes.

The overhead to start the retrieval engine consisted mainly of reading in all document locations (eigenvectors) from disk to a modified kd tree in memory. The kd tree is a data structure for matching data on a large number of keys – in this case, the keys corresponded to the relatively large number of dimensions. Kd trees are a type of multidimensional tree developed by Friedman, Bentley & Finkel (1977), intended to solve the challenge of quick searching of multi-keyed data. This is a similar problem to that of multi-way trees such as B-trees (cf. Knuth, 1998), but whereas B-trees are split on a single key (such as a filename), kd trees are split on multiple keys (such as dimensions in a multidimensional space). The particular memory-based implementation of the kd tree utilized for ISpace was derived from Weiss (1999).

Although up to 300 dimensions were available on disk, as mentioned above, only 100 were used for the tree. Even so, only about 7 million of the 14 million VLC documents were considered for retrieval at one time on the ES-10000, using about 4.5GB of memory. (There was no opportunity for dedicated access to the ES-10000.) Note that only 14 million of the 18 million VLC documents were assigned locations in the information space, because the other 4 million had none of the 3136 terms (many of these were in non-English languages).

Efforts are underway to store the kd tree on disk, rather than storing the entire tree in memory. This is a required development to grow ISpace beyond 7 million documents in 100 dimensions.

The general purpose of the kd tree is to minimize the number of document vectors that need to be compared by brute force (one at a time) to the query. Consider that a key difference between ISpace (and other LSI-like approaches) and the VSM or Boolean systems is that a document can be “close” to a query without having any (or many) terms from the query.

For example, a document, “pig farmer” might be close in ISpace to a query, “swine domesticator.” In order to determine and rank the closest documents to a query, it is necessary to consider ALL documents, not just those with query terms. This is an important drawback of non-orthogonal term vectors. For up to a few thousand documents with coordinates in main memory, an exhaustive comparison and ranking is reasonable. For millions of documents, as in the large Web track, it is desirable to group documents in some sort of structure so that a relatively small proportion of documents need to be exhaustively compared.

There are only two differences between a typical kd tree and the modified kd tree used here (tentatively called a green-black or gb tree for its similarity to a red-black tree). One difference is that there are three children per parent, instead of two. The other is that the decision of which branch or leaf to select at each level is based on the coordinates at the corresponding dimension, instead of on the dimension with the highest remaining variance. These changes are anticipated to be helpful with a highly multidimensional space (hundreds or thousands of dimensions), as compared to the more typical case from the literature on algorithms where only a dozen or so dimensions are used.

- Insertion time for a kd tree is proportional to $n \log(n)$ where n is the number of items to be inserted (i.e., the number of items in the completed tree).
- Cell identification time, to find single cell with the best match, takes time proportional to $\log n$.

- Target identification time, to evaluate all items in a cell (leaf) to find the document vector closest to the query vector, takes linear time with respect to the number of items in the cell.

For ISpace, the goal was to minimize the number of linear time comparisons by minimizing the number of cells that had to be examined. The challenge, as might be expected, is that the tree depth does not nearly reach the total number of dimensions. For example, a tree with 7 million items might reach a depth of 60 levels (that is, the most dimensions examined to separate documents into separate leafs – known as buckets – is 60). This is for a relatively large bucket size of 1000 (a limit of 1000 documents per leaf, before the leaf is split and the tree descends a level).

Smaller bucket sizes would result in deeper levels, but bucket creation and splitting the parent bucket's contents is relatively expensive.

The modified kd tree used for ISpace in TREC8 yielded strong system-based performance, with time per query well under 1 second. Further work with this type of tree structure will determine the extent to which the performance win on search time can also yield good retrieval performance.

Conclusion

This year's implementation of ISpace incorporated query term expansion. A modest-sized information space of only 3136 terms was built, consisting of term eigenvectors derived from a term correlation matrix.

ISpace retrieval performance was not as good for TREC8 as for TREC7. However, the indexing and speed performance was increased by at least an order of magnitude. Implementation of the following should result in regained retrieval performance. All of these features were present in earlier versions of ISpace, but not implemented for TREC8:

1. Allow more terms in the information space. At least 40,000 term eigensystems should be achievable.
2. Incorporate part of speech tagging to identify terms that should be expanded.
3. Implement sentence parsing for TREC-like queries so that terms in phrases identifying unwanted concepts may be bypassed.
4. Multiple options for term weighting schemes
5. Term co-occurrence relations measured at the sub-document level (e.g., within an N-term window or within the same sentence.
6. Add awareness and differential term weighting for SGML/HTML tags such as headings.

Finally, it should be noted that this year's ISpace is an IR system without an interface. Adding a Web-based front end, enabling relevance feedback and an overall higher level of interactivity will present no problems. Incorporating the navigable fly-through system, Yavi (used for TREC7) is also desirable in the near-term.

Post-TREC development has included implementation of the following:

1. Enabling every term to be indexed. Consistently with Witten et al. (1999), simple techniques were developed to insure that the large number of terms with low occurrences

did not take a disproportionate amount of disk space or memory. An index with 982K terms was derived from the AdHoc collection.

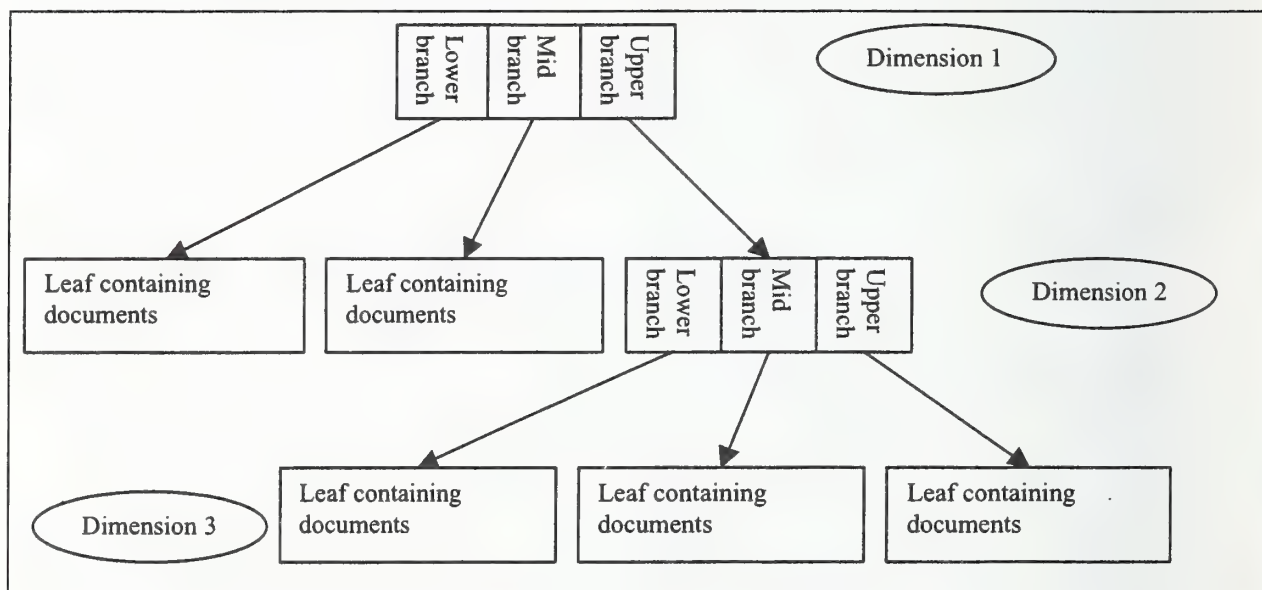
2. More term eigenvectors. An information space with 17,583 unique stemmed terms was computed.
3. Further portability and fewer limitations. ISpace runs under Solaris, Linux and Irix. It uses files of any size permitted by the underlying operating system.

The goals of the TeraScale Retrieval project, as mentioned above, are supported by these ISpace developments. For TREC9, these goals have been identified:

1. To seek higher retrieval performance while maintaining high speed and efficiency.
2. To (re-) incorporate retrieval techniques from prior years.
3. To provide further abstraction in the ISpace system, so that fusion of results from Information Space, VSM, probabilistic and Boolean methods may be achieved.
4. To clarify the relationships among LSI, Information Space, VSM and other methods, to identify their common mathematical themes and seek out opportunities for mutual benefit.

Information retrieval is not a solved problem. ISpace, as described here, is a system that attempts to expand the choice of retrieval techniques available to information scientists, while drawing heavily on prior achievements.

Figure 1: Diagram of gb tree structure. The tree consists of leafs or buckets, which contain documents, and branches, which are used to determine where a document should be placed.



References

- Brill, Erik. (1994). "Some advances in rule-based part of speech tagging." Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94). Seattle, Washington.
- Deerwester, Scott; Dumais, Susan T.; Furnas, George W.; Landauer, Thomas K.; Harshman, Richard. 1990. "Indexing by Latent Semantic Analysis." J. Amer. Soc. For Information Science 41(6): 391-407.
- Frakes, William B. & Baeza-Yates, Ricardo (Eds.). (1992). Information Retrieval Data Structures & Algorithms. Englewood Cliffs, New Jersey: Prentice-Hall.
- Friedman, Jerome H.; Bentley, Jon Louis; Finkel, Raphael Ari. (1977). "An Algorithm for Finding Best Matches in Logarithmic Expected Time." ACM Trans. On Math. Software 3(3): 209-226.
- Knuth, Donald E. (1998). The Art of Computer Programming Volume 3: Sorting and Searching. Reading, Massachusetts: Addison-Wesley.
- Korfhage, Robert R.; Rasmussen, Edit M.; Belkin, Nicholas; Harman, Donna. (1999). "Invitational Workshop on Information Retrieval Tools." Pittsburgh: School of Information Science. (Available online: <http://www.sis.pitt.edu/%7Eerasmus/workshop.html>).
- Witten, Ian H.; Moffat, Alistair; Bell, Timothy C. (1999). Managing Gigabytes. San Francisco: Morgan Kaufmann.
- Newby, Gregory B. (1998). "Context-Based Statistical Sub-Spaces." Text REtrieval Conference (TREC-6) Proceedings, pp. 735-746. Gaithersburg, MD: National Institute of Science and Technology.
- Newby, Gregory B. (1999). "Information Space Gets Normal." Text REtrieval Conference (TREC-6) Proceedings, pp. 567-571. Gaithersburg, MD: National Institute of Science and Technology.
- PITAC (President's Information Technology Advisory Council). (1999). "Information Technology Research: Investing in Our Future." Washington, DC: National Coordinating Office for Computing, Information, and Communications. (Available online: <http://www.ccic.gov/ac/report/>).
- Weiss, Mark Allen. (1999). Data Structures and Algorithm Analysis in C++ (2nd ed.). Reading, Massachusetts: Addison-Wesley.
- Yang, Kiduk & Maglaughlin, Kelly. (2000). "IRIS at TREC8." In this volume.

Retrieval Performance and Visual Dispersion of Query Sets^{1,2}

Mark Rorvig³
The University of North Texas

Abstract

In the course of eight TREC Conferences, retrieval performance of all systems started high and then declined. This was especially true for conference 5. Only in conferences 7 and 8 have performance levels reached those initially achieved. In this paper, scaling of the corpus of 450 TREC topics is performed. It is observed that as the visual dispersion of a topic set increases, the level of retrieval performance across systems declines for that set. Conversely, as the visual dispersion of topics decreases, system performance rises. In common elements of conferences 2, 5, and 8, this relationship appears to hold despite increases in the number of participating systems in TREC. It is proposed that visual dispersion measures should be used to describe topic set difficulty in addition to measures such as "hardness".

Introduction

In the middle of a wonderful review article of the work of Project Intrex from 1965 to 1973, the authors interject this startling phrase: "Our analysis has shown that *choice of words used in search strategies* has a major influence on retrieval effectiveness" (Overhage and Reintjes, 1974, p. 174).

This phrase startles because it is at once a reduction and an enigma. There is no doubt that word choice is important, but how can it be so important that retrieval performance depends upon it to the exclusion of so many other system and architecture considerations? The query is often the last item considered in IR testing. Usually its study is incorporated in the interaction effects between systems and users; a difficult and fluid

arena. The suspicion that queries might establish a system performance limit did not arise in TREC literature until conference 5 (Voorhees and Harman, 1997). However, it has since been recognized as an area for important study, resulting in the establishment of a query track since conference 7 (Buckley, 1998).

It is difficult to quantify the meaning of topic difficulty. Voorhees and Harman (1997) note that it is weakly ($r = 0.33$) correlated with the percent of unique relevant documents for that query. In the same volume, Sparck Jones remarks that "...low levels of performance...in TREC 4 and 5 must be taken as representing a more realistic retrieval situation than TREC 2 and 3..." (Sparck Jones, 1997, p. B-2). Sparck Jones comments further

¹ This study was supported by Intel Corporation.

² A color version of this paper is available at <<http://www.unt.edu/ir/trec/trec8.htm>>.

³ Correspondence should be sent to the author at mrorvig@unt.edu.

in her review of TREC7 that "Since TREC-7 full topics are shorter than TREC-6, but TREC-7 performance levels are better, the TREC-7 topics are presumably not as hard...However, performance is not as tightly correlated with topic length, and specifically with version, as might be expected..." (Sparck Jones, 1999, p. B-6).

Factors that *cannot* describe query difficulty are: (1) topic components (concepts, narratives, etc.), (2) topic length, (3) and topic construction (creating topics without regard to existing documents vs. the contrary practice). Document uniqueness is the only quantitative measure so far offered. Indeed, topic hardness appears to rest in that zone of phenomena that many can mutually observe, but cannot describe in terms that would eventually permit control.

This paper proposes an additional quantitative measure for query difficulty. The measure is applicable to sets of topics only, but is based on the scaled similarity of documents by text terms. The proposed measure is replicable, and conforms to observed system performance behavior across three representative TREC conferences.

Methodology

TREC Topics were copied from the trec.nist.gov site and parsed into individual documents. A document similarity matrix was created using the cosine vector measure of similarity. The similarity matrix was scaled using maximum likelihood method customary for text data (Rorvig, 1999a) and plotted using a conventional graphics tool.

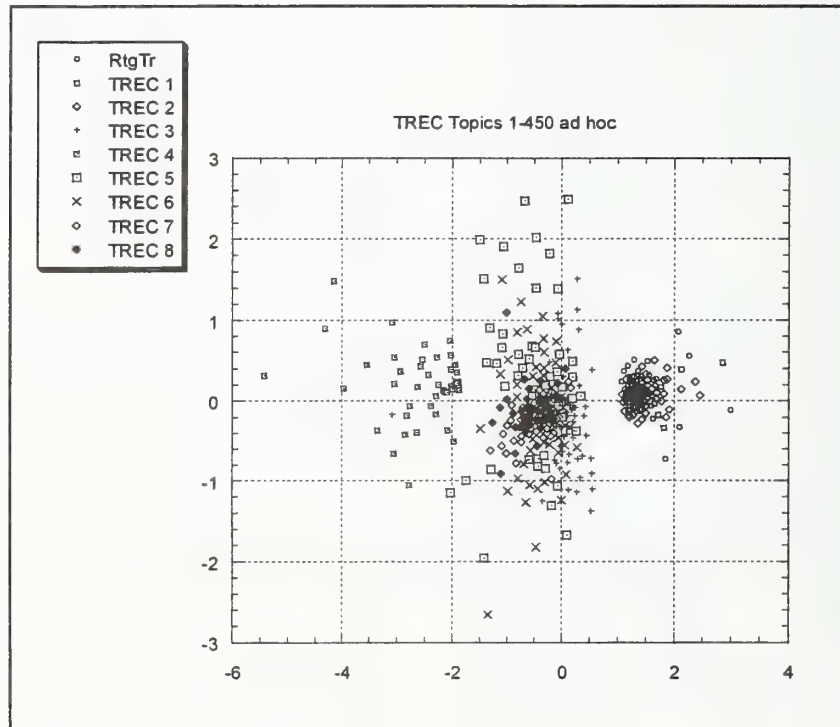


Figure 1: Each dot in the illustration above represents a TREC topic. Arrayed from left to right, topic sets reveal increasing dispersion from topic set 3 onward. This effect does not change until topic sets 7 and 8 appear.

The resulting plot appears as Figure 1. Measures of mean distances among individual documents by set were then taken according to the methodology established in Rorvig and Fitzpatrick (2000). In this method, the centroid of

all points in a set is established, and distances of all other points to the centroid calculated. The mean, standard deviation and minimum and maximum point distances were all calculated. These points appear in Figure 2.

TREC	Mean	Std Dev	Minimum	Minimum
routing	0.3111723	0.2805747	0.0747101	1.6268146
1	0.2052631	0.2294708	0.0091761	1.5116826
2	0.3036261	0.1861311	0.0382413	0.9796236
3	0.5627928	0.4018929	0.0406492	1.7142475
4	0.6950999	0.4838904	0.0755262	2.8757636
5	1.0031172	0.6126555	0.1658590	2.4142985
6	0.7522161	0.4572351	0.1520691	2.6426799
7	0.3396361	0.1987803	0.0501366	0.9060473
8	0.3288653	0.2413289	0.0360555	1.3208982

Figure 2: Calculations of interpoint distances of all TREC topic sets after scaling.

As Voorhees and Harman (1997, p. 18) note regarding the reports by Buckley, Singhal, and Mitra, show a comparison of the average precision of the Cornell runs over five TRECs. "Of particular interest here is the fact that the TREC-5 Cornell system performed about 34% worse on the TREC-5 topics than on the TREC-4 topics...most of this difference is due to 'harder' topics." It is an unusual coincidence that the mean dispersion of TREC-5 topics over TREC-4 topics is, in fact about 31% greater.

Because of the differences in various TREC conferences regarding query construction, the three TRECs with the widest variation in dispersion were chosen for further analysis. From published system reports, ad hoc run

precision scores of participating systems were recorded at 10% levels of recall and 50% levels of recall including manual systems for TRECs 2, 5, and 8.

Results

Figure 3 shows overall system performance for TRECs 2, 5, and 8 for all systems, the top ten systems, and the top twenty-five systems and precision set at 10% and 50% of recall. Although among the top ten systems at high levels of precision, there is no significant difference, significant differences appear for all systems at high and medium levels of recall, medium levels of recall for the top ten systems, and for the top twenty-five systems at both high and medium levels of recall.

TREC	p10/all	p50/all	p10/10s	p50/10s	p10/25s	p50/25s
2	0.481	0.385	0.596	0.353	0.559	0.314
5	0.371	0.177	0.556	0.282	0.498	0.250
8	0.447	0.213	0.595	0.319	0.578	0.307

Figure 3: High and medium precision scores for ad hoc runs for three TRECs of all reporting systems, the top ten systems, and the top twenty-five systems.

Discussion

At issue are the various causes of dispersion among the topic documents. Greater dispersion in scaling can be due to a number of factors, among them simple differences in document length, greater heterogeneity in document tokens, or greater heterogeneity in document tokens among some documents but not in others. Investigation of these factors is beyond the scope of this study at this time, however, they are topics of further interesting exploration.

There is also to consider the unique document theory. Although the correlation reported earlier between document hardness and document uniqueness was not high, there is other evidence that high dispersion among topic statements is reflected in wide separation among their associated documents (Rorvig, 1999b). This would appear to support the document uniqueness theory, and upon replication with the qrels document sets for TREC5 suggest other methods by which document uniqueness and document hardness could be calculated.

Finally, as a point of reference, for the next round of TREC, the topic dispersion more than likely will reflect topic hardness. It will make an interesting postscript to this paper to suggest overall system performance for

TREC9 merely from introducing the scale of the new topics into the similarity matrix calculated for this study.

Conclusions

This paper is an example of thinking with visualization. A correspondence between topic dispersion in a scaled and visualized space and overall TREC system performance was observed based both on previously published statements of TREC participants and direct observations from printed TREC ad hoc run results. It may be possible to predict overall system performance in TREC9 by scaling the topic set when it becomes available.

Acknowledgements

The author is grateful to David Evans of Claritech for suggesting this study and to Ellen Voorhees of NIST for her helpful comments on data used in this paper.

References

- Buckley, C. (1999) "The TREC-7 Query Track," in the *Seventh Text Retrieval Conference (TREC-7)* (E.M. Voorhees, D.K. Harman, Eds.), NIST Special Publication 500-242, p. 65-72.
- Overhage, C.F.J., Reintjes, J.F. (1974) "Project Intrex: A General Review," *Information Storage and Retrieval*, 10:157-188.

Rorvig, M., (1999a) "Images of Similarity: A Visual Exploration of Optimal Similarity Metrics And Scaling Properties Of TREC Topic-Document Sets," *Journal of the American Society for Information Science*, 50(8): 639-651, 1999.

Rorvig, M., (1999b) "On the Orderliness of TREC Relevance Judgements," *Journal of the American Society for Information Science*, 50(8): 652-660, 1999.

Rorvig, M., Fitzpatrick, S. (2000) "Shape Recovery: A Visual Method for Evaluation of Information Retrieval Experiments," *Journal of the American Society for Information Science*, [in press].

Sparck Jones, K. (1997) "Summary Performance Comparisons TREC-2, TREC-3, TREC-4, TREC-5," in the *Fifth Text Retrieval Conference (TREC-5)* (E.M. Voorhees, D.K. Harman, Eds.), NIST Special Publication 500-238, p. B1-B6.

Sparck Jones, K. (1999) "Summary Performance Comparisons TREC-2 through TREC-7," in the *Seventh Text Retrieval Conference (TREC-7)* (E.M. Voorhees, D.K. Harman, Eds.), NIST Special Publication 500-242, p. B1-B6.

Voorhees, E., Harman, D. (1997) "Overview of the Fifth Text Retrieval Conference (TREC5)," in the *Fifth Text Retrieval Conference (TREC-5)* (E.M. Voorhees, D.K. Harman, Eds.), NIST Special Publication 500-238, p. 1-28.

Ask Me Tomorrow: The NRC and University of Ottawa Question Answering System.

Joel Martin
National Research Council
Ottawa, Canada
joel.martin @nrc.ca

Chris Lankester
University of Ottawa
Ottawa, Canada
clank@site.uottawa.ca

Funny how one extra Perl sort function can make a score of 0.52¹ on the question answering task look a lot like 0. This paper describes our brute force approach to the question answering task and how we *did* achieve some success, despite formatting problems with our answer file. This paper also describes how we conducted automatic evaluations using the NIST judgment file on newly proposed answers.

A good question says what it is about and constrains the form of the answer. In other words, it specifies the *distinguishing context* for an answer and partially describes the kind of information that would count as an answer. The question "Who wrote the Declaration of Independence?" specifies that we are talking about someone who wrote a particular document, instead of engaging in some other action with respect to some other document. In addition, the question describes that we are looking for a "who" which must be a person, agency, or institution.

The process of finding an existing answer to a question means finding part of a document that matches the distinguishing context of the question and then extracting an answer of the right type from nearby. Ideally, the extracted answer would have the right sort of relationship to the context of the question.

Since this is our first year at TREC, and we were starting with no existing code, we decided to take a brute force approach to the problem. We divided the task into three phases. Phase I does a very high recall retrieval using the words in the question with the goal of discarding 90% of the document collection. Even with only a tenth of the documents, who wants to read them all? Phase II does an exhaustive scan of all 200-500 byte windows in the retrieved tenth, looking for strings with high similarity to the original question. This phase also ranks these text windows and adds extra points if an obvious answer type is nearby. Finally, Phase III was intended (and did a poor job of it) to extract the best answer of the right answer type from the best outputs from Phase II.

In the rest of this notebook paper, we first describe the three phases in more detail and then describe how we evaluated the system. We end with a discussion of the results and ideas for future work.

1.0 The Question Answering System (QA)

The question answering system was written in Perl and run on several Unix-based (Linux and Solaris) machines. Paragraph indexing and retrieval was

¹ This is the score of our system based on the judgement of (non-NIST) human assessors .

done using an unmodified copy of MG 1.3 (Managing Gigabytes, Witten, Moffat, Bell, 1999). As noted before, the overall system was divided into three largely independent phases, each with a different goal.

1.1 Phase I: High Recall

Even though our system searches for answers in a brute force manner, it cannot exhaustively search through all the documents in the collections. Thus, we want to first select from all the documents those that are likely to contain all the answers. Achieving high recall is often quite difficult because it is hard to keep the list of retrieved documents small. In our case, we do not need the list of retrieved documents to be small yet. That is the task of the later phases. We only care that about 90% of the documents are excluded from the Phase I results.

Figure 1 shows the major substeps involved in performing Phase I. This is a description of the version of Phase I that was used for the answers submitted to NIST. After seeing the NIST test questions, we omitted the use of Wordnet to get synonyms as it was hurting the performance of Phase I.

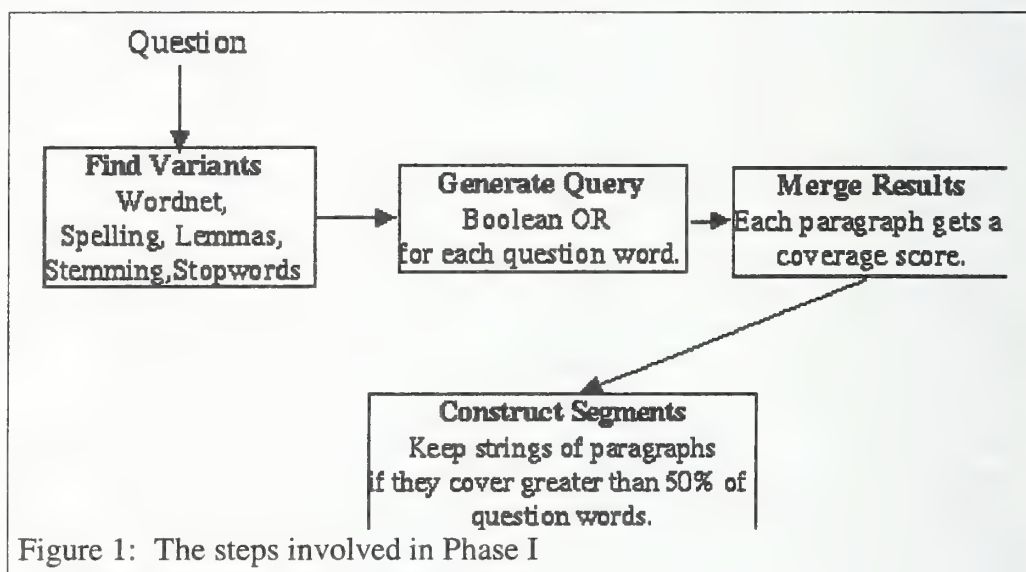


Figure 1: The steps involved in Phase I

In Phase I, each word of the question is first turned into a very long boolean query by disjoining many different forms of the word. For example, the word 'question' would be turned into a query like: "question OR query OR questioning OR questionable OR questoin". Each of these queries is then sent to a search engine (Witten, Moffat, Bell, 1999). This results in one set of documents retrieved for each cloud of related words. In our system, each document is a single paragraph.

These sets of retrieved documents are then merged so that each document has a score reflecting how many of the non-stopwords in the question appear in that document. For instance, if the question were "How are questions turned into queries?", the paragraph immediately above this one would contain the words (or variants thereof): 'questions' and 'queries' and have a score of 2.

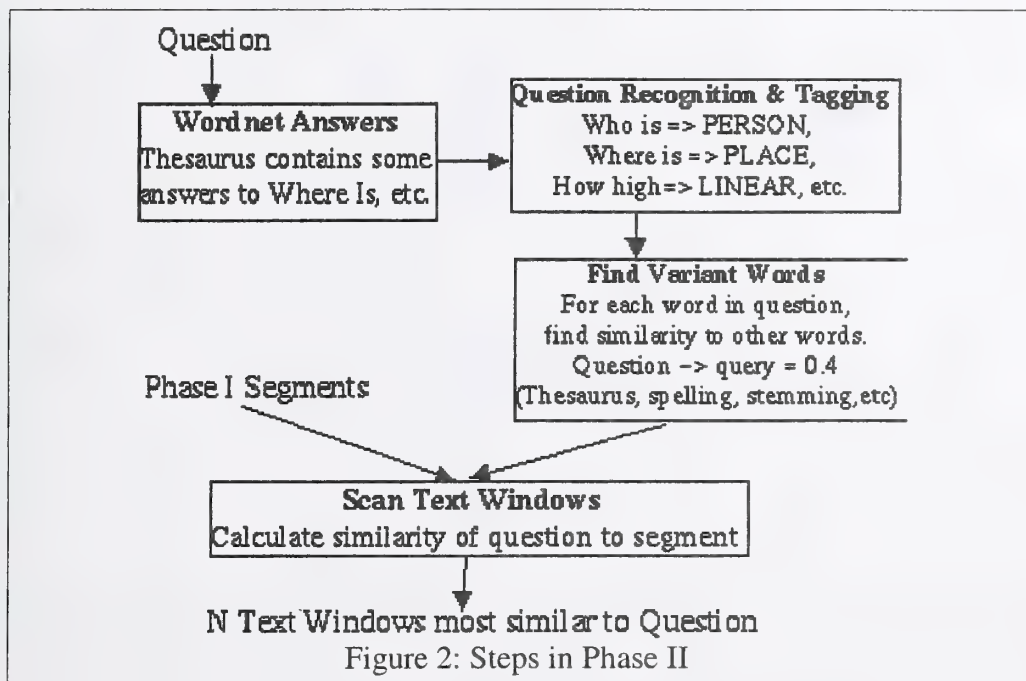
This score is calculated as an estimate of which documents contain more

of the distinguishing context of a question. Finally, those paragraphs which are highly ranked are connected to adjacent high-scoring paragraphs to form segments. Segments are formed because the distinguishing context and the answer to the question may not appear in the same paragraph. Segments and paragraphs are discarded if they contain fewer than half of the question words.

1.2 Phase II: Higher Precision

The QA system assumes that the distinguishing context of a question will appear in a fairly small chunk of text and that the answer will be in close proximity. Thus, after locating those documents that contain the important words, the system should search for strings of words that densely pack those important words. If the words appear close together and possibly in the same order in a given paragraph, then that paragraph is better than another one where the important words are spread throughout the text.

Figure 2 shows the main steps of Phase II. First, Phase II augments the question with known answers stored in the thesaurus and with a tag indicating what type of question it is. Second, it builds a lookup table that stores a similarity score for a word to a word that appears in the question. This lookup table results in a less computationally expensive calculation of the similarities of many text segments to a given question.



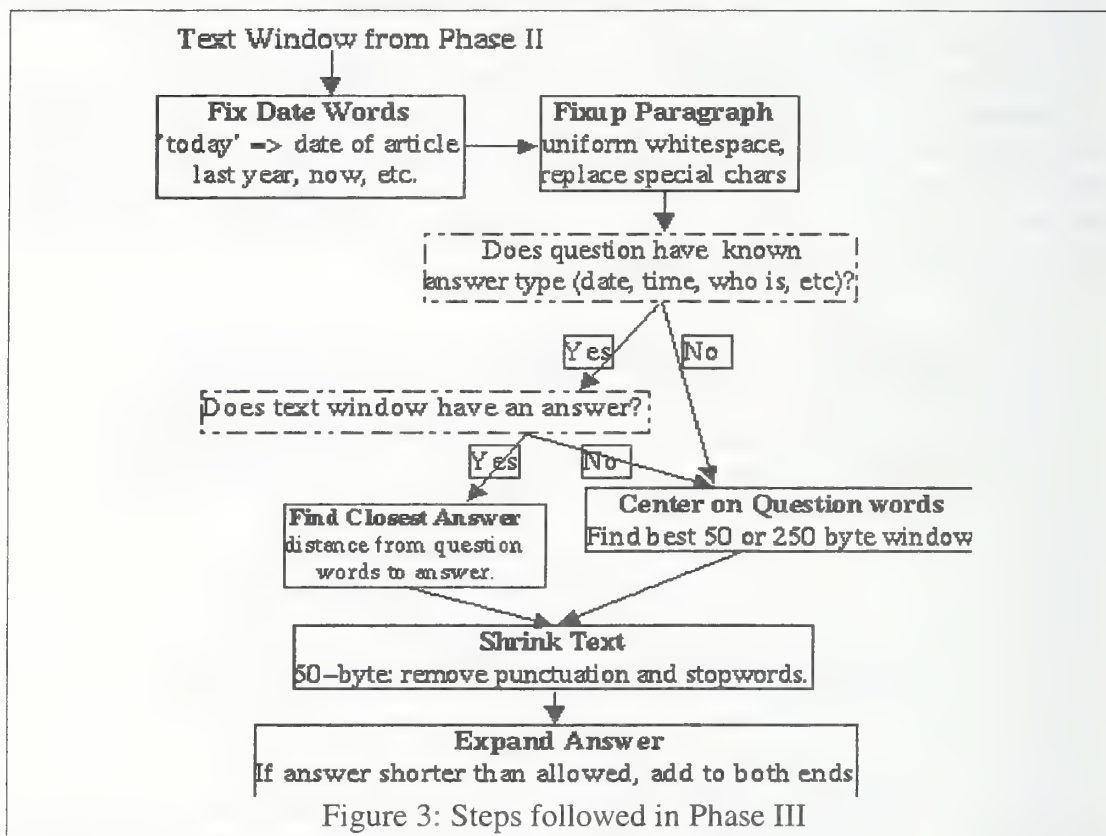
Phase II assumes that there is a definition of similarity between two strings of words that is based on the similarities of the words, their relative position, and the addition or deletion of words. As well, there are several parameters that define the relative impact of each factor on an overall similarity score. This is much like an inverse sort of edit-distance.

Ideally, a large amount of training data would allow setting similarity parameters automatically. However, because only a small amount of training data was available for this competition, the parameters were set by hand.

1.3 Phase III: Answer Extraction

The results of Phase II contain many small windows of text that are very likely to contain the distinguishing context of a question. The task for Phase III is to scan the text in the window or just outside of it, looking for something that looks like an answer to the question. There is no magic algorithm for this phase. There must be a large and growing dictionary of question types and patterns for recognizing answers. If this is true, the success of Phase III should be less a result of the algorithm and more a result of having the right dictionary.

Figure 3 shows the main steps of Phase III. The first two steps of this phase should be done during indexing (and will be next year). This will save processing time and ensure consistent simplification of the paragraphs.



Phase III assumes that there are question types and corresponding answer types. Furthermore, it assumes that the answers are in close proximity to the distinguishing context of the questions. Using a dictionary of question types, it checks whether the question is of a recognized type, such as "Who is" or "Who was". If the question type is recognized, Phase III uses the corresponding pattern to find possible answers, such as "John Smith". When more than one answer is found, the answer that is closest to words that also appear in the question are better.

If, on the other hand, the question type is not recognized, Phase III finds the text window that covers as many words from the question as possible, trying to center on the distinguishing context. In either case, Phase III tries to maximize

the amount of information contained in an answer for a given size of answer. If, for example, the answer must be 50 bytes long: a) the answer loses all its punctuation and stopwords, and then b) is expanded to be as close to 50 bytes as possible by adding tokens on the right and left of the window.

2.0 Using NIST Judgments for Automatic Evaluations

As we fix our QA system and want to test intermediate versions of it, we will not get human evaluations for every set of outputs. Instead, we need some automatic approximation to the official evaluations. This means that we must somehow compare the proposed answer strings from our new system with the NIST answers that appear in the judgment file.

Unfortunately, the NIST answers will not always match the lengths and boundaries of proposed answers. For example, two systems could choose the text window boundaries differently. One system might choose the answer substring: "by Hugo Young (Farrar," whereas another might choose: "Thatcher by Hugo Young". Furthermore, several NIST answer strings have had some tokens removed, such as punctuation, SGML tags, or even some words. As a result, simple string searching will not give a good measure of the correctness of proposed new answers.

As a first pass, we can say that a proposed answer is correct if it fully contains a correct NIST answer. If the NIST answer is correct and a proposed answer contains all of it, the new answer must also contain the correct information. Of course, because of the problems of string matching, we will have to put the answers into a (closer to) canonical form. To do this, we replaced all strings of whitespace and punctuation with a single space. In addition, we replaced all special characters (&) and SGML tags with a single space.

Even with this notion of *contained-within*, there are at least three more problems. First, in the NIST answers, there are several examples where the same substring is deemed both a correct answer and an incorrect answer. For example, in question 54, one correct NIST answer is "54 FBIS4-3997 1 22 April" and one incorrect NIST answer is "54 FBIS4-19846 -1 22 April". This may be because the judges looked at the substring in context and judged one document to contain the answer substring by accident. Second, in at least one case, a substring is correct if capitalized and incorrect if not capitalized: "China" is right, "china" is wrong. Third, although a correct answer may be contained in a proposed answer, it might be hidden among distracting information. For example, the proposed answer may list three proper noun phrases when only one is requested. In such a case, the proposed answer may be judged to be incorrect because the answer is buried.

To partially address these considerations, we augmented the definition of *contained-within*. Some apparent cases of *contained-within* can be *invalid* if the proposed answer also contains an incorrect answer. If this situation arises, we can use the source document to decide whether the proposed answer matches the correct answer or the incorrect one. More concretely (from question 103), suppose a proposed answer, P = "Estonia last month, in which 900 people died. Norway is", contains a correct NIST answer, C = "900". Further suppose that there is an incorrect NIST answer, I = "900 people", that is *contained-within* P

and also contains (or is equal to) C. In this case, our only (automatic) recourse is to consult the source documents. Since the proposed answer came from the same document as the correct answer and not the same document as the incorrect answer, the proposed answer is rated as 'correct'.

```

flag <- contained-within(correct, proposed)
  correct_simp <- replace_punctuation, spaces, SGML, &amp; with space
  proposed_simp <- replace_punctuation, spaces, SGML, &amp; with space
  if (correct_simp is a substring of proposed_simp) then
    flag <- true
  else
    flag <- false
  end if

correct_flag <- isCorrect(P, correct_answers, incorrect_answers)
  correct_flag <- false
  foreach C in correct_answers (while correct_flag equals false)
    if (contained-within(C, P) then
      if (not invalid(P, C, incorrect_answers)) then
        correct_flag <- true;
      end if
    end if
  end loop

invalid_flag <- invalid(proposed, correct, incorrect_answers)
  invalid_flag <- false
  foreach I in incorrect_answers (while invalid_flag equals false)
    if (contained-within(I, proposed) AND
        contained-within(correct, I) AND
        documentOf(I) equals documentOf(proposed)) then
      invalid_flag <- true
    end if
  end loop

```

Table 1: The algorithms for automatic scoring of correct answers.

The specific rules applied are shown in Table 1. These rules are very conservative. They were derived to minimize the chance of rating a proposed answer correct when it is wrong (low false positive rate). It clearly misses many correct answers because there are no correct strings contained in the proposed answer. As in the above example (question 103), there are many correct substrings of "a last month , in which 900 people died. Norway is", but only a select few would be recognized as correct. If the judgment file continues to grow, this problem of many false negatives would gradually improve.

We must remember that the automatic score is generally only a lower bound on the performance of the question and answer system. If System A gets a score of 0.72 and System B gets a score of 0.56, that doesn't necessarily mean that System A is best. It could be that System A did a better job of matching the text windowing used to generate the NIST answers. As a side effect of how the NIST answers were generated, any official QA submission that was scored would receive very close to the same score by this automatic method as was reported by TREC (a very tight lower bound). Hence, we cannot easily use the results of the automatic evaluation to compare a new system

against an official TREC submission.

Even though this metric is only a lower bound on question answering performance, it can still help improve our QA system. The automatic nature of this evaluation would allow it to be run many times in order to, for example, optimize the settings of several parameters. Suppose a system received a score of 0.72 by human evaluation and 0.63 by the automatic evaluation. If you can improve your system so that the automatic evaluation gives a score higher than 0.72, you know that you have a true improvement.

3.0 Results

Although our system's performance is difficult to compare to that of the other question answering systems in TREC, we can use the automatic evaluation to place a rough lower bound on performance. Two additional difficulties arise when doing this with the NIST answers. First, five of the NIST test questions have no answer in the judgments file and as a result, those questions cannot be marked correct. Second, because we are judging an answer correct if a NIST answer is contained-within it, it is very difficult to get an informative lower bound for the 50-byte task, (a 50-byte answer is not likely to contain many other answers).

We evaluated the original 250-byte submission using both the automatic evaluation method described in the last section and human judges. These scores are based on the full set of 198 questions, even though we are guaranteed to get 5 of them incorrect. Table 2 shows the results for the three phases.

	<i>At least one answer in output</i>	<i>NIST Rank Score</i>
<i>Phase I</i>	73%	---
<i>Phase II</i>	68%	---
<i>Phase III</i>	52%	0.4
<i>Human</i>	59%	0.52

Table 2: Results of the three phases for the answers submitted to NIST, August 1999

We have also re-run the first 100 questions after examining the test questions and answers and removing most of the sensitivity to synonyms. These results are shown in Table 3. All three phases improved markedly. In particular, Phase I now has 98 questions with at least one answer in the result list. Phase II is moderately better, as is Phase III. The (over) use of WordNet seemed to be a bad idea for the TREC test questions.

	<i>At least one answer in output</i>	<i>NIST Rank Score</i>
<i>Phase I</i>	98%	---
<i>Phase II</i>	81%	---
<i>Phase III</i>	67%	0.54

Table 3: Results of the three phases after removing WordNet.

Phase III is still a loose collection of heuristics and will require some organization to support further improvements. Many answers are lost because they appear one paragraph earlier than the identified answer or are lost because the date substitution is done incorrectly.

4.0 A Final Word

We are happy with our moderate success this year. Most of the product of our efforts is experience, and that will be applied in future competitions. As such, we hope that the QA track continues.

One way that the QA track might mature is to better define the types of questions that are included in TREC and those that are excluded. This year there was some process that selected appropriate questions from among submitted questions. Was this process random or where some questions omitted because their answers were not of the right form or they were difficult to find with typical search engines?

As noted in our discussion of Phase I, we originally designed the recall step to be robust to the use of synonyms in questions. Indeed, many of our ten submitted questions assumed some ability to recognize synonyms. Oddly enough, our system was quite good at handling this sort of question which did not appear in the TREC test questions.

Over the next few months, we expect to improve our question answering system. Besides optimizing the system to handle the TREC test questions, we will be generating a larger set of training questions. This should allow us to find more question-types and the corresponding answer-types.

We also want to speed up Phase I and Phase II. We will attempt to do this with as few minor changes to an existing search engine, as is possible. The goal will be to replace the function of Phase I and Phase II, while at the same time taking advantage of existing data structures in a search engine.

Phase III needs the most improvement. This phase attempts to extract an answer of the right type from paragraph-sized windows of text. Improvement in this phase may require a huge knowledge-engineering effort to find and characterize different types of queried information. This effort may require reproducing some of the effort that went into many of the systems designed for the MUC conferences.

5.0 Acknowledgments

The authors wish to thank Terry Copeck, Rob Holte, Ken Barker, Stan Matwin, and Stan Szpakowitz for discussions and assistance with questions,

answers, and the document collections.

This project is funded by an NSERC/NRC Research Partnerships grant:
NSERC-NRC #653-022-96 "Intelligent Information Access"

6.0 References

Witten, I. H., Moffat, A. & Bell, T. C. (1999). Managing Gigabytes: Compressing and Indexing Documents and Images. San Francisco: Morgan Kaufmann.

Using Coreference in Question Answering

Thomas S. Morton

Department of Computer and Information Science

University of Pennsylvania

`tsmorton@linc.cis.upenn.edu`

1 Introduction

In this paper we present an overview of the system used by the GE/Penn team for the the Question Answering Track of TREC-8. Our system uses a simple sentence ranking method which is enhanced by the addition of coreference annotated data as its input. We will present an overview of our initial system and its components. Since this was the first time this track has been run, we made numerous additions to our initial system. We will describe these additions and what motivated them as a series of lessons learned after which the final system used for our submission will be described. Finally we will discuss directions for future research.

2 Initial System Overview

The input to our system is a small set of candidate documents and a query. To get a set of candidate documents we employed a search engine over the TREC-8 document collection and further processed the top 20 documents returned by it for each query. In order for our system to annotate coreference relations, a variety of linguistic annotation is required. This includes accounting for SGML tags in the original documents, performing sentence detection, tokenization, noun phrase detection, and named-entity categorization. With this annotation complete, the coreference system annotates coreference relations between noun phrases. The coreference-annotated document is then passed to a sentence ranker which ranks each of the sentences, merging these ranked sentences with the sentences from previously processed documents. Finally the top 5 sentences are presented to the user.

2.1 Search Engine

In order to get a small collection of candidate documents, we installed and indexed the TREC-8 data set with the PRISE 2.0 search engine, developed by NIST. Indexing and retrieval were done using the default configuration with no attempts made to tune the ranking to the Question Answering task. From this we took the top 20 ranked documents and performed further processing on each of them.

2.2 Preprocessing

Determining coreference between noun phrases requires that the noun phrases in the text have been identified. This processing begins by preprocessing the SGML to determine likely boundaries between segments of text, sentence-detecting these segments using a sentence detector described in (Reynar and Ratnaparkhi, 1997), and tokenizing those sentences using a tokenizer described in (Reynar, 1998). The text can then be part-of-speech-tagged using the tagger described in (Ratnaparkhi, 1996), and finally noun phrases are determined using a maximum entropy model trained on the Penn Treebank (Marcus et al., 1994). The output of Nymble (Bikel et al., 1997), a named-entity recognizer which determines which words are people's names, organizations, locations, etc., is also used to aid in determining coreference relationships.

2.3 Coreference

Once preprocessing is completed, the system iterates through each of the noun phrases to determine if it refers to a noun phrase which has occurred previously. Only proper noun phrases, definite noun phrases, and non-possessive third person pronouns are considered. Proper noun phrases are determined by the part of speech assigned to the last word in the noun phrase. A proper noun phrase is considered coreferent with a previously occurring noun phrase if it is a substring of that noun phrase, excluding abbreviations and words which are not proper nouns. A noun phrase is considered definite if it begins with the determiner "the" or begins with a possessive pronoun or a past-participle verb. A definite noun phrase is considered coreferent with another noun phrase if the last word in the noun phrase matches the last word in a previously occurring noun phrase. The mechanism for resolving pronouns consists of a maximum entropy model which examines two noun phrases and produces a probability that they co-refer. The 20 previously occurring noun phrases are considered as possible referents. The possibility that the pronoun refers to none of these noun phrases is also examined. The pair with the highest probability are considered coreferent, or the pronoun is left

unresolved when the model predicts this as the most likely outcome. The model considers the following features:

1. The category of the noun phrase being considered as determined by the named-entity recognizer.
2. The number of noun phrases that occur between the candidate noun phrase and the pronoun.
3. The number of sentences that occur between the candidate noun phrase and the pronoun.
4. Which noun phrase in a sentence is being referred to (first, second, ...).
5. In which noun phrase in a sentence the pronoun occurred (first, second, ...).
6. The pronoun being considered.
7. If the pronoun and the noun phrase are compatible in number.
8. If the candidate noun phrase is another pronoun, is it compatible with the referring pronoun?
9. If the candidate noun phrase is another pronoun, is it the same as the referring pronoun?

The model is trained on nearly 1200 annotated examples of pronouns which refer to or fail to refer to previously occurring noun phrases.

2.4 Sentence Ranking

Sentences are ranked based on the sum of the *idf* weights (Salton, 1989) for each unique term which occurs in the sentence and also occurs in the query. The *idf* weights are computed based on the documents found on TREC discs 4 and 5 (Voorhees and Harman, 1997). No additional score is given for tokens occurring more than once in a sentence. If a sentence contains a coreferential noun phrase then the terms contained in any of the noun phrases with which it is coreferent are also considered to be contained in the sentence.

A secondary weight was also used to resolve ties in the first weight ranking and to determine how sentences longer than 250 bytes should be truncated. The secondary weight was computed for each noun phrase based on the sum of the *idf* weights for each unique term where words occurring farther away from the noun phrase were discounted. This was done by adding the product of the *idf* weight for a word and the reciprocal of the distance, in words, between the noun phrase and the word. For example, a word three tokens to the left of a noun phrase would only receive a third of its *idf* weight with respect to that noun phrase. This weight was used to select a "most central" noun phrase and the weight of this noun phrase was used to resolve ties between

sentences equally ranked by the first score. In cases where a sentence was longer than 250 bytes, this noun phrase was used to determine where the sentence would be truncated.

3 Lessons Learned

3.1 Lesson 1

Our first goal was to develop a baseline with which we could compare our system's output. The simplest baseline we could imagine would be to simply rank segments of text based on the common *tf · idf* measure. Since these segments were small, having a maximum of 250 bytes, we ignored the term frequency component, and the query and segment were treated as a set of terms rather than a bag. Each segment was ranked based on the sum of the *idf* weights for the words in that segment which, once stemmed, matched those found in the query. Each segment was a 250-byte window centered on a term which was also found in the query. On the development set provided, this produced an answer in the top five sentences for nearly half (17/38) of the questions provided for development. This allowed us to better assess the added value various types of linguistic annotation would provide.

3.2 Lesson 2

Performing linguistic annotation of the documents in the collection is computationally expensive. While we only examined the top 20 returned documents, some of these documents were very long, often exceeding 2MB. To combat this, each document was reduced to a 20K segment using the 250-byte segment ranked first by the baseline as the center of the 20K segment. This sped up processing considerably but had no noticeable effect on system output. This may be because some question generation was based in part on reading the documents and creating questions which were answered by that document. This may have lead to a bias for shorter documents.

3.3 Lesson 3

Many questions indicate a semantic category that the answer should fall in based on the Wh-word the sentence uses. For Wh-words such as "Who", "Where", and "When", a fairly specific category is specified while the category for "What" and "Which" is usually specified by the noun phrase following it. "How" can be used to specify a variety of types; however, when it is followed by words such as "many", "long", "fast", the answer will likely contain a number of some sort. When the semantic type of the answer could be determined, and this could be mapped to a category that was determined by the named-entity recognizer or some other recognizable pattern, then only sentences which evoked an entity of the same category were considered. This

included sentences which contained pronouns which referred to entities of the correct type in preceding sentences. Sentences which contained the correct entity type, but all entities of this type were present in the query, were also ignored. This processing helped exclude sentences which only used terms in the query and would be highly ranked even though they did not contain a possible answer.

3.4 Lesson 4

The semantic category for questions which ask for a date can usually be determined. These are also categories that the named-entity recognizer identified and so the system was quite effective at finding candidate answers to these sorts of questions. However, the form of the answer often did not meet the needs of the user. Within the context of a newspaper article, relative date terms such as today, Tuesday, last week, or next month, can be interpreted by a reader based on context; however, when this context is removed, the meaning of these terms is often unclear. All the articles in this collection contain datelines which often make it possible to automatically resolve such terms for the user. For these terms we used the dateline as a base reference for when the article was written and then used a small set of heuristics to determine a complete description of a date term. Additional terms introduced by the more complete description of the relative date term were also considered to be in that sentence. This was especially helpful when these terms were in the query and would not have matched this sentence without such processing. This processing was also helpful when presenting sentences to the user. When sentences contained relative date terms, the parts of the description which were not present in the relative date term were inserted after it to improve the user understanding of the text without context.

3.5 Lesson 5

While linguistic processing is helpful in determining answers to a variety of questions, some information needs can be satisfied with much simpler means. For questions asking "Where is X" or "What is the capital of X", a good online dictionary will usually provide the answer within a few keystrokes. For these two types of questions, we automatically extracted a set of probable answers and added these to the query. This improved system performance and did a better job of addressing the user's intentions than the system without this information. Specifically, the dictionary provided answers with a better level of generalization than the system did without these additional query terms.

4 Final System

Our final system examined the query and added terms from an online dictionary when applicable.

This expanded query was then passed to the search engine, and the top 20 documents returned by it were collected for further processing. The baseline system was run on these documents to find a central passage, and a 20K window around this passage was kept for further processing. Preprocessing was performed on these segments and coreference relations between entities and dates were automatically annotated. Finally, sentences which weren't excluded by the semantic-category filtering were ranked using the simple *idf* weighting described above. The top-ranked sentences were augmented to include complete descriptions of coreferential terms such as definite noun phrases, proper nouns, pronouns, and dates, which were not already present in the sentence. These augmented sentences were then presented to the user.

5 Results

The part of the TREC-8 Question Answering Track evaluation in which we participated allowed 5 answers to be submitted, each of which could be at most 250 bytes long. For the 198 questions in the evaluation, our system was able to answer 126 of them, or 63.3%. If answers are weighted by rank, our mean reciprocal rank was 0.510. This compared favorably with other systems; of the 20 participants, our system ranked 4th overall.

6 Discussion and Future Work

Attending TREC-8 provided us with additional insights for future work. The most significant of these is that in the future more attention needs to be paid to indexing. Specifically, we discovered that the search engine we used, PRISE 2.0, was significantly below the state-of-the-art in performance at the ad-hoc task. To compare its performance at the Question Answering task, we considered all the documents in which some participant had found a correct answer. This is likely not the complete set of documents which contain the answer, but it serves as a reasonable approximation. We then compared the number of these documents that PRISE found compared to AT&T's search engine. The result is that the AT&T search engine returned 146 more documents, over all queries, in which some system found the answer than the PRISE search engine. For 38 queries, the PRISE system returned no documents in which an answer was found, while the AT&T system did this for only 35 documents. We should also explore the possibility of examining more than 20 documents. This is evidenced by the fact that if all 200 documents returned by the AT&T system are considered, then a document containing an answer was provided for at least 187 of the 198 queries. In a similar vein, we also hope to look at alternate indexing schemes such as paragraph indexing, which was

used in Southern Methodist University's system.

7 Conclusion

Here we present a system for performing question answering on a large collection of text. This system uses a simple ranking method, which is aided by determining coreference relations to add terms to a sentence and by determining the semantic category of the answer to exclude some sentences from consideration. We believe coreference plays an important role in question answering, as it allows a system to extract answers from text which refers to but doesn't explicitly mention an entity. It also provides a means to make text presented to the user without its original context easier to understand. Determining the semantic category that the answer will be in, and the entities which fall into that category, is also useful: it allows sentences which do not contain a possible answer to be excluded from consideration. This system performed well at the evaluation and we look forward to improving its performance for future evaluations.

References

- D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*.
- M. Marcus, B. Santorini, and M. Marcinkiewicz. 1994. Building a large annotated corpus of english: the Penn Treebank. *Computational Linguistics*, 19(2):313-330.
- Adwait Ratnaparkhi. 1996. A Maximum Entropy Part of Speech Tagger. In Eric Brill and Kenneth Church, editors, *Conference on Empirical Methods in Natural Language Processing*, University of Pennsylvania, May 17-18.
- Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16-19, Washington, D.C., April.
- Jeff Reynar. 1998. *Topic Segmentation: Algorithms and Applications*. Ph.D. thesis, University of Pennsylvania.
- Gerald Salton. 1989. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Publishing Company, Inc.
- Ellen M. Voorhees and Donna Harman. 1997. Overview of the fifth Text REtrieval Conference (TREC-5). In *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, pages 1-28. NIST 500-238.

Interactive Okapi at Sheffield - TREC-8

M. Beaulieu, H. Fowkes, N. Alemayehu and M. Sanderson

*Department of Information Studies
University of Sheffield
m.beaulieu@sheffield.ac.uk*

Abstract

The focus of the study was to examine searching behaviour in relation to the three experimental variables, i.e. searcher, system and topic characteristics. Twenty-four subjects searched the six test topics on two versions of the Okapi system, one with relevance feedback and one without. A combination of data collection methods was used including observations, verbal protocols, transaction logs, questionnaires and structured post-search interviews. Search analysis indicates that searching behaviour was largely dependent on topic characteristics. Two types of topics and associated search tasks were identified. Overall best match ranking led to high precision searches and those which included relevance feedback were marginally but not significantly better. The study raises methodological questions with regard to the specification of interactive searching tasks and topics.

1. Experimental objectives and setting

The University of Sheffield's participation in the Interactive Track is a continuation of the work initiated at the very outset of TREC at City University based on the Okapi system. With respect to the stated high level goal of the Interactive Track in TREC-8, which is to examine the process as well as the outcome, the Sheffield experiment focused principally on the process. The aim was to investigate interactive information seeking behaviour and user perceptions of the retrieval process using two versions of the highly interactive Okapi IR system, one with relevance feedback and one without relevance feedback. The specific objectives were threefold, each relating to the different experimental variables, i.e. searcher, system and task, as follows:

- to examine information seeking patterns of behaviour and determine how behaviour is shaped by the characteristic of the task and the functionality of the system;
- to determine how the different interactive searching features of the Okapi system namely, the best-match ranking, best-passage retrieval and incremental query expansion facility impacted on searching behaviour;
- to consider how searcher perceptions of the searching task are supported by the functionality of the interface.

The same configuration of the Okapi system was used as in TREC-6 and -7. A full description is found in (1). Searchers were subjected to two experimental conditions over the six topics. Each of the 24 searchers performed three searches on the system with relevance feedback and three on the system without relevance feedback, with 144 searches being carried out in total.

1.1 Data collection methods

In order to capture the multiple dimensions of the interactive searching process for qualitative analysis, i.e. searcher/topic, searcher system and topic/system interactions, other data collection methods were used in addition to the standard Interactive Track questionnaires. The test instruments included:

Observations: a structured approach was adopted to enable the experimenter to record the search process in four stages corresponding to the retrieval sub-tasks, i.e. search formulation and reformulation, viewing and evaluating results.

Transaction logs: the systems' extensive logging facility provided quantitative data on search interactions complementing the qualitative observational data.

Verbal protocols: searchers were instructed to 'think aloud' as they interacted with the system in order to get some insight into their perceptions, problems, strategies and understanding of the task in hand. The protocols were also used to gain a better understanding of any inconsistencies that emerged between the observational and interview data.

Questionnaires: four types of questionnaires common to all participants in the Interactive Track were administered by the experimenter. The pre-session questionnaire established searcher skills and experience. The post-search questionnaires ascertained the level of familiarity and ease/difficulty of the six individual topics. The post-system questionnaire gathered information on the ease of use and learnability of the two versions of the system. The final post-session questionnaire collected data on searcher preferences and views of the experimental conditions.

Interviews: following the standard post-search questionnaires, additional more probing questions were asked in order to gain more insight into searchers' perceptions of the individual topics and search tasks. A final post-session semi-structured interview provided further information on the system's interactive search features as well as the overall experimental session.

2. Searching behaviour

2.1 Query formulation

In over half of the searches, subjects formulated initial search queries by simply extracting keywords from the given topic descriptions. The single exception was for the Tropical Storms topic where two thirds of searchers also generated their own query terms. It appeared that there was some ambiguity with this topic. Some searchers interpreted it as searching for different types of storms, e.g. hurricanes, typhoons, as indicated in the topic description, whilst others were looking for actual named tropical storms.

Overall the norm was to enter between two and four single query terms which corresponded to the number of keywords in the actual topic descriptions (Table 1). The highest number of terms were entered for Tropical Storms and Tourism Violence, the reason being in part because more keywords appeared in the topic itself.

Table 1. Initial number of query terms entered

No query terms	1	2	3	4	5	6	7	Total no searches
Total no searches	7	35	50	26	14	9	3	144
	5%	24%	35%	18%	10%	6%	2%	100%

2.2 Query reformulation

Overall queries were reformulated for just over half of the searches carried out on both versions of the system. There was little incentive to modify a query if searchers were still finding instances of the required information in initial results, as for example for the topics on Birth Rates, Robot Technology and Tourism. Likewise, searchers were more likely to modify an initial query when they were finding few relevant documents. This was the case for Tropical Storms, Cuba Sugar and Tourism Violence, where a higher number of negative relevance judgements were made in relation to the total number of items viewed (Table 2, 3). Generally there was a strong correlation between the number of negative relevant judgements and the number of iterations in a search session.

Table 2. Number of negative relevance judgements

No. non relevant documents	0- 5	6-	11-	16-	Total no. searchers
Tropical storms	3	10	7	4	24
Cuba sugar	2	5	12	5	24
Birth rates	10	10	3	1	24
Robots	20	4	-	-	24
Tourism	11	11	2	-	24
Tourism violence	8	10	4	2	24
Total no searches	54	50	28	12	144

Table 3. Number of positive relevance judgements and saved documents

No. saved documents	0-5	6-	11-	16-	Total no. searchers
Tropical storms	4	18	2	-	24
Cuba sugar	18	6	-	-	24
Birth rates	-	10	14	-	24
Robots	2	14	8	-	24
Tourism	-	7	13	4	24
Tourism violence	11	9	4	-	24
Total no searches	35	64	41	4	144

With respect to the use of relevance feedback for query modification, there was no evidence to show that the availability of the relevance feedback facility encouraged searchers to reformulate queries compared to searches carried out on the system with no relevance feedback. Half of the searches undertaken on the system with relevance feedback were reformulated. Out of the 36 reformulated searches, in 17 cases the queries were expanded without any modification to the term list presented in the working query, which in effect can be considered as a form of automatic query expansion. For the remaining 19 searches, where users manipulated the candidate list of terms for query expansion, they were more likely to add and experiment with their own query terms than to experiment with those suggested by the system. This was particularly true for the three topics where it was difficult to identify relevant documents.

2.3 Viewing document hitlists

Three aspects of searching behaviour were examined in relation to how searchers viewed and selected documents from hitlists. Firstly, we consider the relationship between document rankings and items saved. Secondly, we compare how far down searchers worked through an original hitlist before modifying or stopping a search, and thirdly whether searchers worked through the hit lists in a comprehensive or selective manner.

The system displays the top fifty documents from each retrieved set. In 85% of searches more documents were viewed and saved from the top 25 items than from the bottom half of the ranked list. Moreover in 28% of searches documents were saved only from the top 25. Whilst this may imply agreement between the searcher and the system, some aspects of the experimental design may have created a bias towards more documents being saved from the top of the list. Firstly searchers were instructed to ignore documents which duplicated information already saved. Hence this resulted in the exclusion of relevant lower ranking documents. The design of the interface also made it difficult to substitute documents previously saved with an item found further down the ranking. The time limit would also have prevented searchers from exploring the full list.

How far searchers actually worked down the ranked list varied a great deal across the six topics. In half of the searches subjects went as far down as the 30th item in the ranked list. However for the Tropical Storms topic they were more likely to go to the bottom of the list. This is possibly explained by the number of duplicate material which occurred and could be skipped over. Duplicate material was also a feature of the

Tourism Violence topic, but by contrast over half of the searchers did not go beyond the 30th ranking. In this case the display of the term occurrence information provided an effective indicator of document relevance. Searchers could assume that if the keyword 'Tourism' did not appear, the document was unlikely to be relevant. However for the Tropical Storms topic the display of query terms associated with an individual item was less helpful and it was difficult to judge promising documents.

In another topic, Robot Technology, two-thirds of searchers only examined items from the top 20 documents in the list. This was largely due to the time searchers took to view and engage with the documents. In some cases there were a number of 'instances' to be considered in an individual document and more time was required to read the content and get to grips with the topic.

Searchers adopted two strategies for working through a hitlist. They either viewed documents sequentially or were more selective. For topics where there appeared to be a high level of potentially relevant items, e.g. Robot Technology, Tourism, Birth Rate, the tendency was to view each item in turn whereas for topics where there appeared to be fewer relevant items, e.g. Tropical Storms, Cuba Sugar, Tourism Violence, searchers were more likely to skip through and be highly selective.

2.4 Viewing full documents

2.4.1 Passage retrieval

On viewing a document the searcher is taken to a highlighted 'best' passage which represents the section of the document which scores the highest in relation to query term occurrence. The function of the highlighted best passage is not only to assist users in determining relevance, but it also serves as a source for extracting terms for query expansion and the searcher is given the choice to make a relevance judgement on the full document or the passage only. An analysis of searching behaviour in relation to passage retrieval was undertaken for three of the topics where the feature came most into play, i.e. Birth Rates, Robot Technology and Tourism. Documents related to the other topics were generally much shorter, and the 'best passage' was not an option.

Searchers inevitably started by scanning or reading the information in the highlighted passage. However in over half of the searches, subjects examined other parts of the document before making a relevance judgement based on the passage only. It appeared that searchers sought more contextual information and evidence outside the highlighted passage before making a relevance judgement. Alternatively the motivation was also to look for additional instances outside the highlighted passage. In the case of the Tourism topic searchers were more likely to make a judgement on the basis of the passage only. Searchers were in effect looking for the right combination of labels. If a sentence include the keywords 'tourism', 'increase' and a number, the document was deemed to be relevant. However on the whole searchers were not confident in making relevance judgements on best passages only. The verbal protocols also revealed that the passage only option was perceived as a means of making a weak relevance judgement, i.e. an indication that the whole document was only partially relevant.

2.4.2 Scanning vs reading

Scanning was the most prevalent strategy for evaluating the contents of documents. This was in part dictated by the time limit for the search task and the need to find as many different instances as possible. Query term highlighting was essential for document viewing. However scanning and reading were usually carried out simultaneously. Inevitably searchers had to supplement scanning with reading in order to establish contextual relationships between query terms. For example the terms 'Tourism' and 'Violence' retrieved documents on the impact of political violence on tourism as well as those on violence targeted directly against tourists. Hence the level of engagement with the topic had a major influence on the procedural level of interaction. When labels were explicit (e.g. names, numbers) searchers would extract instances by plotting highlighted keywords. For topics such as finding the latest developments and applications of Robot

Technology, which was more cognitively demanding, scanning keywords did not provide enough clues. Other factors which contributed to the number of searchers who read as the primary activity for viewing documents included the density of the 'instances' found in the document as well as the lack of familiarity with the topic.

2.4.3 Interpreting topics

Overall searchers had little familiarity with the topics, Birth Rates being the most familiar and Cuba Sugar and Robot Technology being the least familiar. As searches were carried out subjects expressed different levels of certainty regarding the task in hand. Through the behavioural observations and the verbal protocols it was possible to identify different cognitive states and levels of engagement associated with the different topics. The descriptions below provide some insight into how searchers interacted conceptually with the topics on Tropical Storms, Cuba Sugar, and Robot Technology.

Tropical Storms

The different interpretations for this topic seemed to have an influence on the levels of certainty and patterns of searching behaviour. Searchers who interpreted the question as finding different types of storms ended up questioning and doubting their search goal as the search progressed. They made more negative relevance judgements than those searchers who understood the task to be looking for names of tropical storms.

The more common interpretation of the topic was to search for the names and locations of different types of tropical storms. This led to a different level of engagement with the topic as well as uncertainty. Some of the searcher-system interaction for this topic was very superficial involving pinpointing names of storms. In fact 6 out of the 24 subjects never read any of the documents at any point in the search. Others by contrast adopted a deeper level of engagement with the topic, treating the property damage/loss of life as a separate criteria that documents had to fulfil in order to consider a document as being relevant. This resulted in another level of questioning and doubting as searchers were unsure whether to save a document if the evidence of property damage/loss of life was only implied. A typical example of this was the dilemma over whether it was possible to infer property damage from references to 'heavy landslides' or damage to 'communication infrastructure'. A high number of searchers engaged either in trying to predict whether a document met the relevance criteria or in trying to avoid documents from the hitlist which were concerned with a specific storm already identified.

Robot Technology

For this topic most searchers expressed a high level of uncertainty about the content of the information found in the initial stages of the search. Although there were a few searchers who adopted a common search pattern of scanning for names of applications, one distinctive feature of the searching behaviour was that searchers attempted to comprehend and make sense of the documents. This involved a great deal of reflection and interpretation whereby information was explicitly extracted, compared, categorised and summarised. This type of conceptual activities and reasoning was largely absent or was not evident from the searching behaviour associated with the other topics. Some searchers even attempted to grapple with the conceptual complexities of defining the latest developments and differentiating between current and future developments.

Cuba Sugar Imports

In contrast with the other topics, there was generally a high level of certainty about what to look for during the initial search process. This however gave way to doubt and uncertainty in the viewing/evaluation stage owing to the few relevant documents found in the database. A significant number of searchers thus changed their searching strategy in the course of the search. Two different approaches were adopted. Some searchers reformulated their query which resulted in a more positive state, as the subsequent duplication of named countries provided confirmation that there were only a few countries involved. Others however came to a

similar conclusion but the uncertainty was resolved by a deeper level of engagement with the topic. By reading and comprehending the actual text, searchers assimilated information about the broader economic context and the trade embargo and were able to confirm that there were only a few answers to the topic request.

3. User perceptions of search tasks

Users experienced little difficulty in starting searching namely because as mentioned previously they simply extracted keywords from topic descriptions. Three quarters of searches were deemed to be easy or somewhat easy. Tourism and Birth Rates were the easiest and Cuba Sugar, and Robot Technology were classed as the most difficult. It appears that an inherent feature of a difficult topic is the level of familiarity and the understanding of the content and context of the issues. For example the technological content of the Robot technology topic and the economic or geographical issues addressed in the Cuba Sugar topic made it more difficult for the searchers to absorb the text. Another indicator of a difficult topic was the higher level of engagement or effort required from searchers. The Robot Technology topic comprised two elements, to find the latest developments as well as applications. Whilst for the Cuba Sugar topic, the task involved finding hidden labels and there were many false leads before finding a correct 'instance'.

Perceptions of search ease/difficulty are closely linked with search satisfaction. Hence for the topics perceived as more difficult (Cuba Sugar and Robot Technology) searchers were more reluctant to rate search satisfaction very highly because they were uncertain of the scope of the topic. Search satisfaction is also a product of the quality of what was found rather than how much is found. Searchers expressed the lowest level of satisfaction for the Robot Technology topic even though overall they found the highest number of instances related to it.

In all of the topics, excluding tropical storms, two-thirds of searchers were very or quite confident that they had identified all of the instances. In the case of Cuba Sugar and Tropical Storms they were more confident because there appeared to be few instances anyway. There is also a clear relationship between how many instances are identified and searcher perceptions of whether or not they had enough time to do an effective search. For topics where the most instances were found three quarters of searchers said that they didn't have enough time. Conversely for topics where more duplicate material was found, searchers indicated that they did have enough time to carry out the search. The results also reinforce the relationship between searcher confidence and the amount of time needed. In topics where searchers were most confident that they had identified all the instances, they were also likely to claim that they had enough time.

4. User perception of the system

There was little difference in the perception of the ease of use or learnability between the two versions of the system. Both systems were deemed to be easy. It seems that the difficulties that searchers had in manipulating the working query did not colour the overall perceptions of ease of use or learnability. This is possibly because after the first or second attempts most searchers abandoned the working query and treated the experimental system in exactly the same way as the control system.

4.1 Relevance feedback and query expansion

Two-thirds of searchers professed a high level of understanding of both systems. However given that three-quarters did not perceive any differences between the two, this might suggest that they did not understand the underlying relevance feedback mechanism and did not readily link the terms of the working query to the retrieved set of documents. Two-thirds of searchers declared a preference for the control system without relevance feedback. The most frequently cited reason given was related to the manipulation of the working query. Searchers confirmed that removing suggested terms individually was a very time-consuming process which didn't warrant the effort. However the desire for more control over the search process was a deeper concern. It appeared that searchers were happy for the system to provide suggestions as long as it presented the candidate terms in a way that didn't threaten to change the direction of the search.

4.2 Best match ranking

Searchers expressed confidence in the ranking of the hitlist as an overall guide to potential relevant documents but they also recognised that the ranking was not a guarantee that items would be relevant. The query term occurrence information was also considered to be useful in providing clues to potential relevancy.

4.3 Passage retrieval

Passage retrieval was perceived as an efficient way of identifying instances and minimising user effort for extracting relevant information. However for topics less familiar to the searchers and where they needed to understand the context of the document, jumping to the highlighted passage was deemed to be procedurally disorientating and counter intuitive. This generated reservations about the suitability and reliability of best passage retrieval in terms of a starting point as well as giving the searcher some indication about how far into the document they were.

At a more conceptual level about two thirds of searchers were sceptical about the support that passage retrieval offered for making positive relevance judgements. In some ways passage retrieval added to the cognitive burden and uncertainty in making relevance judgements. The dilemma was clearly expressed by one searcher as follows:

"All of it is relevant, but then what I actually want to know is in the passage, but then I only know that because I've looked at it all so it's difficult to decide whether I want to say it's fully or partly relevant".

5. Search outcomes

In this section we present the search outcomes as perceived by the searchers as opposed to the actual results or system performance in terms of precision and recall measures.

Table 4 shows that the average number of instances found by subjects for all the searches across the different topics for each of the experimental conditions were comparable, 12.0 instances per search for the system with relevance feedback and 11.7 for the version without relevance feedback. Equally the average number of instances for the searches which were *not* reformulated under both conditions was similar 13.8 and 14.0 respectively. However for searches which were reformulated, the average number of instances decreased significantly to 10.2 instances for searches using query expansion based on relevance feedback and 9.9 for searches not using relevance feedback.

Clearly, the conditions which led to reformulation differed for the different topics. Reformulation was beneficial for three of the topics which required a some degree of interpretation, e.g. Tropical Storms, Cuba Sugar, Tourism and Violence, whereas it appeared less fruitful for the seemingly more straight forward topics where the answer to the request was clearer, e.g. declining Birth Rates, Robot Technology developments, and increases in Tourism. It also appears that query modification is more likely to improve searches with initial poorer results than those which are already successful.

Although search queries which were expanded by using relevance feedback retrieved marginally more instances than search queries which were reformulated by searchers generating their own additional query terms, queries which were in effect expanded automatically (i.e. searchers accepted all the candidate terms presented by the system in the working query) compared to those expanded interactively (i.e. searchers added their own query terms to the working query), led on average to more instances, 11.2 as opposed to 9.3 (Table 5). Automatic query expansion (AQE) also seemed to be more effective for the topics where instances could be more easily identified (Birth Rates, Robot Technology, Tourism), whereas interactive query expansion (IQE) appeared to be as more productive for more complex topics (Tropical Storms, Cuba Sugar).

Table 4. Instances retrieved with relevance feedback on and off

Condition No. instances	Relevance feedback on			Relevance feedback off		
	All searches	Searches not reformulated	Reformulated searches	All searches	Searches not reformulated	Reformulated searches
Tropical storms	142	54	88	135	54	81
Cuba Sugar	77	26	51	108	29	79
Birth rates	173	97	76	156	91	65
Robots	221	165	56	196	137	59
Tourism	167	138	29	170	109	61
Tourism violence	87	20	67	83	29	54
Total no. instances	867	500	367	848	449	399
No. searches	72	36	36	72	32	40
Average no. instances	12.0	13.8	10.2	11.7	14.0	9.9

Table 5. Instances retrieved with automatic and interactive query expansion

Condition	AQE	IQE
Tropical storms	13	75
Cuba sugar	12	39
Birth rates	64	12
Robots	41	15
Tourism	17	12
Tourism violence	44	23
Total no. instances	191	176
No. searches	17	19
Average no instances	11.2	9.3

6. Search results

The comparative performance of searches undertaken with the system which included relevance feedback as opposed to those on the system without relevance feedback shows that the former led to marginally better precision but the difference is not significant (Table 6). An analysis by individual topics reveals the poorest overall performance for Tourism/Violence where searchers had difficulty in filtering out duplicates. Nevertheless, searches reformulated on the system with relevance feedback did lead to better results than those reformulated without relevance feedback. By contrast the Cuba sugar topic was the second worst in terms of precision but achieved the highest recall. It was perceived as difficult with many duplicates leading to negative relevance judgements and few instances were found. The best precision was achieved for the most straightforward topic on increase in Tourism. Although the Robot Technology topic required more effort and engagement, it led to the second best precision results. Both of these topics required little query reformulation.

Table 6. System Performance with and without relevance feedback

System condition	Precision	Recall
With RF	0.759	0.352
Without RF	0.728	0.393

7. Summary and conclusions

The experiment was primarily concerned with searching behaviour in the use of a highly interactive retrieval system and determining the interaction between the different variables, i.e. searcher, search task/topic and system characteristics, within the context of the TREC Interactive Track experimental framework.

7.1 Searcher characteristics

The twenty-four test subjects were quite a homogeneous group and there appeared to be no evidence of any significant individual differences in searcher characteristics in terms of their familiarity with the topics or the searching strategies they adopted. Moreover search results reveal no significant difference in search performance between individual searchers.

7.2 Topic and task characteristics

Searching behaviour was however largely determined by the nature of the different search topics themselves. Two types of topics and associated search task emerged. On the one hand some topics were clearly defined and instances could be easily identified from the outset. The search task for those topics involved scanning documents to spot keywords, e.g. names of countries, which made it possible for relevance judgements to be readily made. Other topics on the other hand were more complex, requiring some element of interpretation and they were also characterised by a degree of uncertainty. In such cases identifying instances was not simply a question of finding keywords, but relevance was more dependent on establishing the context in which the keywords appeared. Hence searchers had to read the documents, engage with the content and deliberate. Deliberation or uncertainty was largely concerned with defining the scope of the topic from the documentary evidence, e.g. whether or not a named storm was a tropical storm or not. Uncertainty also emerged when few instances could be found.

7.3 System characteristics

Overall the system best match ranking was effective in producing high precision searches rather than searches with high recall. Relevance feedback came into play in different ways depending on the type of topic. Automatic query expansion could improve results of simple straightforward topics whereas for more complex topics, interactive query expansion with contributions from both the searcher and the system appeared to be more effective. Displaying query term information in the hit list as well as highlighting best passages and query terms in documents, assisted users in making relevance judgements for the simple topics but they were less helpful on their own for the more complex topics where searchers had to engage with the content.

7.4 Experimental design and conditions

The results of the study raises methodological questions with regard to the specification of the interactive task and the topics. From a system's perspective it could be argued that the TREC interactive task of finding as many different instances on a topic as possible in twenty minutes is basically a recall task. However from a searcher's perspective it appears to be perceived as a precision task with the emphasis being on finding 'new evidence' not just more evidence. Having the system discard duplication or documents which covered known evidence was the most frustrating element of the task. Thus more attention needs to be given to consider other types of retrieval tasks which may be more appropriate for evaluating interactive searching.

The choice of topics also influenced the nature of the search task. Although the experiment included only six topics, which made it feasible to increase the number of test subjects, fruitful data was collected on the characteristics of topics. More research however is required not only in identifying different types of search topics, but also in defining more closely what constitutes a simple and more complex topic and determining how the different elements should be taken into account in the experimental design. In order to deepen our understanding of interactive searching and its evaluation, a typology of search topics needs to be developed.

References

1. Beaulieu, M. M. and Gatford, M. J. Interactive Okapi at TREC-6. In: Voorhees, E. M. and Harman, D. K. (eds). *The Sixth Text Retrieval Conference (TREC-6)*. Gaithersburg, MD: NIST, 1998, 143- 167.

THE THISL SDR SYSTEM AT TREC-8

Dave Abberley (1), Steve Renals (1), Dan Ellis (2) and Tony Robinson (3,4)

(1) Department of Computer Science, University of Sheffield, UK

(2) ICSI, USA

(3) Department of Engineering, University of Cambridge, UK

(4) SoftSound, UK

Email: {d.abberley, s.renals}@dcs.shef.ac.uk, dpwe@ICSI.Berkeley.EDU, ajr@softsound.com

ABSTRACT

This paper describes the participation of the THISL group at the TREC-8 Spoken Document Retrieval (SDR) track. The THISL SDR system consists of the realtime version of the ABBOT large vocabulary speech recognition system and the THISLIR text retrieval system. The TREC-8 evaluation assessed SDR performance on a corpus of 500 hours of broadcast news material collected over a five month period. The main test condition involved retrieval of stories defined by manual segmentation of the corpus in which non-news material, such as commercials, were excluded. An optional test condition required retrieval of the same stories from the unsegmented audio stream. The THISL SDR system participated at both test conditions. The results show that a system such as THISL can produce respectable information retrieval performance on a realistically-sized corpus of unsegmented audio material.

1. INTRODUCTION

The TREC-8 test collection was obtained from the TDT-2 corpus and consisted of 902 shows (502 hours) of US broadcast news material covering the period from February to June 1998. The collection contained 21754 individual news items (389 hours of material) with the task being to retrieve the set of stories relevant to each of 50 queries. Two retrieval conditions were specified:

Story Boundary Known (SBK) The SBK runs used a corpus which had been segmented manually into individual news stories, with non-news material being excluded. The definition of non-news material for this purpose included fillers as well as commercials.

Story Boundary Unknown (SBU) The SBU runs reflected the more realistic situation where story boundary information is not known *a priori*. Each news broadcast was to be treated as a continuous audio stream and it

was the task of the retrieval system to find the location of the news stories contained within it.

The TREC-8 SDR track was designed to test how SDR systems perform with a much larger document collection than they have been evaluated on previously — the TREC-7 SDR track used only 87 hours of broadcast audio data (2866 stories) [1]. A particular concern was that speech recognition errors would become more dominant as corpus size increased: this problem would be aggravated by a rising out of vocabulary rate caused by the language model becoming progressively out of date over the duration of the corpus. Another concern was to observe the effect automatic segmentation of the corpus would have on retrieval performance.

The THISL¹ spoken document retrieval system consists of the 'real time' version of the ABBOT large vocabulary continuous speech recognizer [2] and the THISLIR text retrieval system [3]. ABBOT is used to transcribe broadcast audio material into text which can be indexed and retrieved by THISLIR. The ABBOT transcriptions can be produced in the order of real time on standard hardware. THISLIR can index and retrieve both segmented and unsegmented news broadcasts.

2. ABBOT SPEECH RECOGNITION

ABBOT is a hybrid connectionist/HMM system [4] which estimates the posterior probability of each phone given the acoustic data at each frame. This differs from traditional recognizers which estimate the *likelihood* that a phone model generated the data. Posterior probability estimation is performed by a set of recurrent networks [5] trained to classify

¹THISL is an ESPRIT Long Term Research project with the objective of developing a spoken document retrieval system which integrates speech recognition, natural language processing and text retrieval technologies. The main goal of the project is to develop a UK English system suitable for a BBC newsroom application. The TREC SDR evaluation provides an ideal framework to evaluate the performance of the system on a closely related task.

This work was supported by ESPRIT Long Term Research Projects THISL (23495) and SPRACH (20077).

phones. Direct estimation of the posterior probability distribution using a connectionist network is attractive since fewer parameters are required for the connectionist model (the posterior distribution is typically less complex than the likelihood) and connectionist architectures make very few assumptions on the form of the distribution. Additionally, this approach enables the use of posterior probability based pruning [6] and is able to provide useful acoustic confidence measures [7]. Decoding is performed by the CHRONOS decoder [8].

THISL produced two sets of speech recognition transcripts for the TREC-8 corpus:

- S1 The S1 transcripts were produced by the ABBOT 'real time' system which was used by the SPRACH consortium in the 1998 DARPA/NIST Hub-4 Broadcast News evaluation [2].
- S2 The S2 transcripts were produced with an improved acoustic model obtained by merging the acoustic probabilities from the 'real time' system with those produced by an acoustic model using modulation-filtered spectrogram features [10]. These transcripts were not produced in time to be used as an official entry in the evaluation but the results obtained with them have been included here as a contrast condition.

2.1. ACOUSTIC MODELLING

2.1.1. S1 REAL TIME SYSTEM

The acoustic model used by the ABBOT real time system consists of two recurrent networks (RNNs) which estimate *a posteriori* context-independent (CI) phone class probabilities. The phone set contains 54 classes, including silence. One network is used to estimate the phone posterior probability distribution for each frame given a sequence of 12th order perceptual linear prediction features [9]. The other network performs the same distribution estimation but with features presented in reverse order, since recurrent networks are time-asymmetric. The probability streams produced by the two RNNs are averaged in the log domain to produce a final set of probability estimates. The models were trained using the 104 hours of broadcast news training data released in 1997.

2.1.2. S2 SYSTEM

The acoustic model for the S2 system was obtained by log domain merging of the probability estimates produced by the RNNs used in the S1 system with those produced by an acoustic model using modulation-filtered spectrogram features [2].

Modulation-filtered spectrogram (MSG) features were developed to be a representation of speech recognition that

is robust to the signal variations caused by reverberation and noise [10, 11]. The robustness is obtained by using a signal-processing strategy derived from human speech perception.

The MSG acoustic model used an MLP containing 8000 hidden units trained on all 200 hours of broadcast news training data downsampled to 4 kHz bandwidth.

2.1.3. LANGUAGE MODELLING

The same backed-off trigram language model [2] was used by both the S1 and S2 systems. Approximately 450 million words of text data was used to generate the model, using the following sources:

- Broadcast News acoustic training transcripts (1.6M words),
- 1996 Broadcast News language model text data (150M),
- 1998 North American News text data:
LA Times/Washington Post (12M), Associated Press World Service (100M), NY Times (190M).

The models were trained using version 2 of the CMU-Cambridge Statistical Language Model Toolkit [12] using Witten-Bell discounting.

The recognition lexicon contained 65432 words, including every word that appeared in the broadcast news training data. The dictionary was constructed using phone decision tree smoothed acoustic alignments [2].

A *fixed* language model and lexicon constructed from material pre-dating the acoustic data were used throughout the evaluation.

3. TEXT RETRIEVAL

3.1. THISLIR

The THISLIR information retrieval system used for TREC-8 is essentially a "textbook TREC system", using a stop list, the Porter stemming algorithm and the Okapi term weighting function. Specifically, the term weighting function $CW(t, d)$ for a term t and a document d given in [13] was used:

$$CW(t, d) = \frac{CFW(t) * TF(t, d) * (K + 1)}{K((1 - b) + b * NDL(d)) + TF(t, d)}. \quad (1)$$

$TF(t, d)$ is the frequency of term t in document d , $NDL(d)$ is the normalized document length of d :

$$NDL(d) = \frac{DL(d)}{\overline{DL}}, \quad (2)$$

where $DL(d)$ is the length of document d (ie the number of unstoppped terms in d). $CFW(t)$ is the collection frequency weight of term t and is defined as:

$$CFW(t) = \log \left(\frac{N}{N(t)} \right) \quad (3)$$

where N is the number of documents in the collection and $N(t)$ is the number of documents containing term t . The parameters b and K in (1) control the effect of document length and term frequency as usual.

3.2. QUERY EXPANSION

If a relevant document does not contain any of the query terms, then the overall query/document weight (computed using (1)) will be 0, and the document will not be retrieved. This can be a particular problem in spoken document retrieval, owing to the existence of recognition errors, and out-of-vocabulary (OOV) query words. *Query expansion* addresses this problem by adding to the query extra terms with a similar meaning or some other statistical relation to the set of relevant documents.

If words are added to a query using relevant documents retrieved from a database of automatically transcribed audio, then there is the danger that the query expansion may include recognition errors [14]. One way to avoid this problem is through the use of a secondary corpus of documents from a similar domain that does not contain recognition errors. For a broadcast news application, a suitable choice for such a corpus is contemporaneous newswire or newspaper text. A query expansion algorithm may then operate on the relevant documents retrieved from the secondary corpus.

Rather than using a blind relevance feedback approach to query expansion that maintains the term independence assumption which underlies the probabilistic model used for retrieval, we have adopted a method based on the consideration of term co-occurrence. Specifically, we have employed a simplified version of the *local context analysis* (LCA) algorithm introduced by [15]. The query expansion weight $QEW(Q, e)$ for a potential expansion term e and a query Q , across a set of R (pseudo) relevant documents is defined as:

$$QEW(Q, e) = CFW(e) \sum_{t \in Q} CFW(t) \sum_{i=1}^R TF(e, d_i) \cdot TF(t, d_i). \quad (4)$$

This approach does not consider distractor (non-relevant, but retrieved) documents. A discriminative term may be included by computing a similar QE weight over a set of distractor documents, combining with (4) using a method such as the Rocchio formula (reviewed by [16]). Experiments have indicated that adding such a discriminative term has a negligible effect. The QE weight (4) is used for ranking potential expansion terms only. Additional weighting can take the form of scaling (1) by $1/\text{rank}$.

The query expansion corpus contained about 25 million words and 36000 news stories from the following text sources:

- TREC-7 broadcast news reference transcripts from June 1997 to January 1998 (0.75M words)

- LA Times/Washington Post texts from September 1997 to April 1998 (14.9M words)
- NY Times texts from January 1998 to June 1998 (odd days only) (9.4M words)

After some development work on TREC-7 data, all experiments added a maximum of 15 expansion terms. The manual segmentation of the QE corpus into stories was retained, rather than employing an automatic segmentation into fixed length passages. Development work indicated that there was no significant difference between the schemes, in terms of average precision, but the manual segmentation resulted in an order of magnitude fewer documents to index.

3.3. AUTOMATIC SEGMENTATION

One of the problems which arises when building a practical news on demand application is that radio and TV news recordings do not contain explicit information about when individual news items begin and end, and so some sort of automatic segmentation scheme is required. Segmentation can be attempted at different stages of processing:

1. *Prior information* from programme scripts, etc. can be used if such material is available from the broadcaster. This information is likely to be incomplete and can't allow for the dynamic nature of a live news broadcast, such as when a new story breaks during the programme.
2. *Acoustic information* can also be used to segment a news broadcast. It is possible to detect periods of non-news such as silence, music [17], adverts [18], etc and exclude these from the material to be decoded. Segmentation at this stage has the additional advantage of reducing the amount of material to be decoded (which can be extremely time-consuming when it is non-speech) and reducing the amount of spurious material to be indexed.
3. *Speech recognition transcriptions* are lists of recognized words together with their start and end times. This information can also be used in the segmentation process.

In TREC-8, the only available prior information was the close caption text of the reference transcripts, and the rules of the evaluation forbade its use. No acoustic segmentation was tried due to lack of time for development work. Whilst this decision led to a decoding overhead, experiments on the TREC-7 evaluation suggested that retrieval performance was unlikely to be hit drastically [19]. Consequently, the THISL system in TREC-8 used an automatic segmentation scheme which relied solely on the information provided by the speech recognition transcriptions.

Following the work of Smeaton *et al* [20, 21], a series of experiments were conducted on the TREC-7 dataset using rectangular windows of various lengths and varying degrees of overlap [19]. Window lengths measured in both time and number of words were tried. Relatively short windows worked best but there was not much performance difference between word and time windows. The latter, however, enjoy the advantage of ensuring each document has a maximum time duration and so THISL used 30 second windows with a 15 second overlap for the SBU runs. The document length normalization parameter b was set to zero.

3.3.1. DOCUMENT RECOMBINATION

Each broadcast was segmented into a set of overlapping documents of 30 seconds duration which were then indexed by THISLIR. These documents obviously bore little relation to the actual news stories which would have been obtained by hand segmentation of the corpus, and this created an additional problem. For scoring purposes, a document was identified in terms of a characteristic time within a broadcast, and it was considered to be *relevant* if that characteristic time fell within the time period (defined by manual segmentation) covered by a *relevant* news story. One of the problems with the THISL segmentation scheme is that adjacent overlapping segments are likely to produce similar scores, causing the list of retrieved documents to contain several segments from the same news item. Any such *additional* documents would be scored as *irrelevant*, and so a document recombination and rescoring scheme was devised.

A simple scheme was used. For each query, the top-scoring 4000 documents were retrieved initially. Any documents from the same broadcast which overlapped each other were recombined into one larger document *provided* that their retrieval rank differed by no more than 200 positions. This *200 rule* was introduced to try and prevent low scoring documents from the same broadcast containing 'random hits' of, say, one relatively unimportant query word from being included in the recombined document. The value of 200 was arrived at by conducting a series of tuning runs on the TREC-7 evaluation set which was redecoded (including the non-news portions) for development work. The optimal recombination threshold is likely to be highly corpus and task dependent and merits further empirical examination: a scheme making use of term weighting would also be worth investigating.

The problem of how to rescore the combined documents was also investigated experimentally. Several schemes were tried including using the maximum score from the set of documents to be combined:

$$W_{\max} = \max_i^n w_i \quad (5)$$

reestimating the Okapi score for the combined document (updating CFW, but not accounting for the overlap between adjacent documents), and other methods with less obvious theoretical justification. On the TREC-7 development data, the best performing rescoring formula proved to be:

$$W_{\text{DERB}} = \frac{\sum_i^n w_i}{1 + (n-1)\frac{d}{t}} \quad (6)$$

where W is the retrieval score for the combined document, w_i is the original score for document i , n is the number of document segments to be combined and t and d are the window length and overlap respectively. This formula — known locally as the *DERB factor* — was arrived at somewhat accidentally² for our UK English system [22] and has the effect of boosting the score of a combined document relative to that of a standalone document. It does not require term frequency information to obtain the new score, and hence can be implemented by post-processing the raw retrieval output.

Subsequent experiments on TREC-8 evaluation data showed that using the maximum score from the set of documents to be combined produced an improvement in average precision (see Section 4.2.3).

3.3.2. BROADCAST SEGMENTATION

The THISL automatic segmentation can be summarized as follows:

1. Entire news broadcast decoded into a *stream* of text.
2. Text stream broken into *documents* using a fixed length rectangular window of 30 seconds with a 15 second overlap.
3. Resulting documents are indexed by THISLIR.
4. At retrieval time, the 4000 top-scoring stories are retrieved. Overlapping documents from the same show are combined into *stories* if retrieval rank difference < 200.
5. Retrieval score adjusted for each story using Equation 6.

3.4. PARAMETER SETTINGS

A locally developed 379 word stop list and the Porter stemming algorithm were used.

The term weighting parameter settings for the SBU and SBK runs are given in Table 1. The SBK parameters have been changed slightly from their TREC-7 settings owing to the larger query expansion database and as a result of experience with our UK English system [22]. Note that the

²Thanks to Sue Johnson for pointing this out!

Parameter	SBK	SBU
b	0.7	0.0
K	1.5	1.5
QE-b	0.5	0.5
QE-K	0.25	0.25
QE-nt	15	15
QE-nr	10	10

Table 1: Parameter settings for TREC-8 SDR SBK and SBU runs.

document length parameter b was set to zero for the SBU run because the automatic segmentation scheme inherently performs approximate document length normalization (see Section 3.3.1).

3.5. QUERY PREPARATION

The text queries were preprocessed before being input to THISLIR by removing punctuation, converting to lower case and expanding numbers and abbreviations/acronyms to make the query more similar to speech recognizer output (eg, 1998 \rightarrow nineteen ninety eight, G-7 \rightarrow G seven).

4. EVALUATION RESULTS

4.1. SPEECH RECOGNITION

Nominal word error rates (WERs) were estimated on a 10 hour subset of the corpus. The ABBOT S1 system produced a WER of 32.0%, and the S2 system improved this to 29.2%. The 'real time' S1 decodings were produced in approximately $3 \times$ real time on a variety of standard hardware. Note that this is the overall *average* figure and the decoding speed of a given broadcast will vary with machine performance. Further, entire news shows were decoded, and the speed of the search phase will have been compromised due to the decoder having to transcribe material such as commercials which it had not been trained on. Table 2 gives a breakdown of the time taken for the different stages of decoding. The S2 system was slightly slower due to the extra processing involved. Dedicated hardware was used at the S2 acoustic modelling stage.

No multiwords or phrases were used in the recognition or retrieval process. OOV words were not a significant problem; as usual, there was one OOV word in the TREC queries (*Filo*), together with a text processing problem (*II's* (as in "Pope John Paul II's") was not expanded to "the second's").

Process	\times real time	
	S1	S2
Feature extraction	0.1	0.2
Acoustic modelling	0.2	0.5
Search	2.8	2.8
Total	3.1	3.5

Table 2: Average time taken at stages of the decoding process.

4.2. INFORMATION RETRIEVAL

4.2.1. STORY BOUNDARY KNOWN (SBK) CONDITION

Table 3 shows the results for the SBK runs with THISLIR for the different sets of transcriptions. Average precision is seen to decrease slightly as Word Error Rate (WER) increases (Figure 1). The S2 run did not produce an improvement in average precision relative so S1 despite the improved WER.

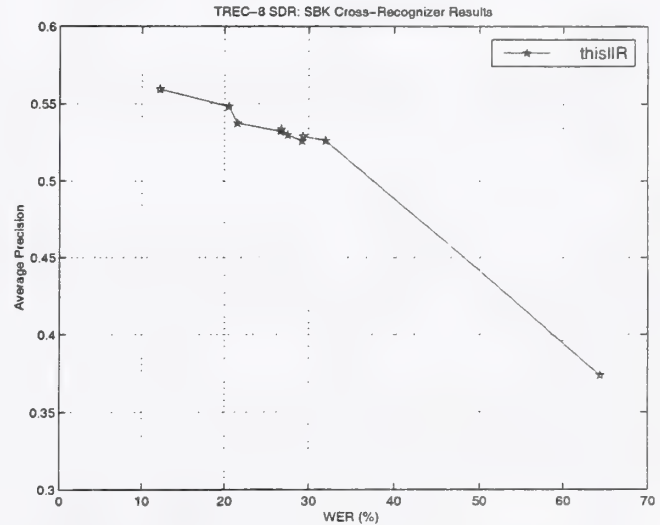


Figure 1: thisIR: SBK Average Precision as a function of WER

4.2.2. STORY BOUNDARY UNKNOWN (SBU) CONDITION

Table 4 shows the results for the SBU runs. Once again, average precision tends to decrease as WER increases. The improved error rate of the S2 run produced a 1% improvement in average precision. Figure 2 illustrates the trend graphically.

Average precision for the SBU runs is about 10% lower (in absolute terms) than for the corresponding SBK runs. Although this is a considerable loss of performance, the

SBK Run	WER	Retrieved	AveP
shef-r1	12.2%	1653	0.5596
shef-cr-cuhtk-s1	20.5%	1638	0.5484
shef-cr-limsi-s1	21.5%	1613	0.5375
shef-cr-cuhtk-s1p1	26.6%	1621	0.5322
shef-b2	26.7%	1587	0.5335
shef-b1	27.5%	1590	0.5298
shef-s2	29.2%	1609	0.5260
shef-cr-att-s1	29.3%	1622	0.5290
shef-s1	32.0%	1594	0.5262
shef-cr-cmu-s1	64.4%	1299	0.3735

Table 3: Summary of results for Story Boundary Known condition. *WER* is word error rate, *Retrieved* is the number of relevant documents retrieved out of a total of 1818, *AveP* is the average precision.

information retrieval capability of the system is still quite respectable with, on average, over 50% of the top 10 documents retrieved being relevant. This is an encouraging result at this stage in the development of automatic segmentation schemes.

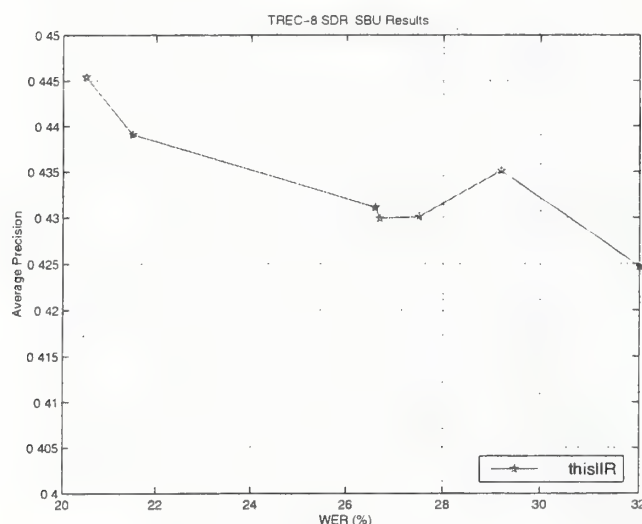


Figure 2: thisIR: SBU Average Precision as a function of WER

4.2.3. EFFECT OF RESCORING METHOD

Table 5 compares the DERB (Equation 6) and MAX (Equation 5) rescoring methods (see Section 4.2.3). On the TREC-8 evaluation set, MAX rescoring gives up to a 1% increase in average precision and a small decrease in the number of relevant documents retrieved³. It is interesting to note that the increases vary depending on the speech recognition tran-

³The reverse was true on the TREC-7 data used for tuning experiments.

scripts used (see Figure 3). There is obviously much scope for experimentation to find the optimum rescoring formula.

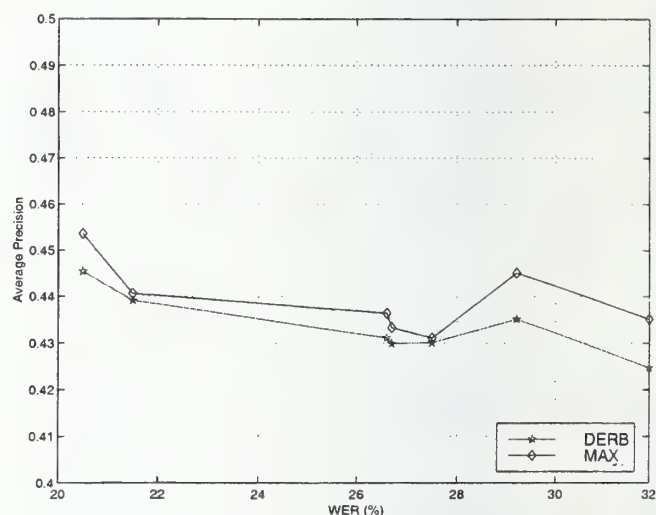


Figure 3: thisIR: SBU Average Precision variation with rescoring method

4.2.4. EFFECT OF NON-NEWS MATERIAL

The segmentation procedure described in Section 3.3 made no attempt to exclude non-news material. It is interesting to estimate what effect this had on information retrieval performance. Table 6 compares the performance of the THIS-IR SBU system on the shef-s1 and shef-slu transcripts. The shef-s1 run represents the 'perfect case' where all non-relevant material (and *only* non-relevant material) has been removed. The figures show that indexing up non-relevant material such as commercials, and also fillers, causes a relatively modest 1.3% loss in average precision. This suggests that most of the performance loss associated with auto-

SBU Run	WER	Retrieved	AveP
shef-cru-cuhtk-slu	20.5%	1458	0.4454
shef-cru-limsi-slu	21.5%	1442	0.4391
shef-cru-cuhtk-s1plu	26.6%	1455	0.4311
shef-b2u	26.7%	1386	0.4299
shef-b1u	27.5%	1393	0.4301
shef-s2u	29.2%	1418	0.4351
shef-s1u	32.0%	1393	0.4247

Table 4: Summary of results for Story Boundary Unknown condition. *WER* is word error rate, *Retrieved* is the number of relevant documents retrieved out of a total of 1818, *AveP* is the average precision.

SBU Run	WER	DERB		MAX	
		Retrieved	AveP	Retrieved	AveP
shef-cru-cuhtk-slu	20.5%	1458	0.4454	1443	0.4536
shef-cru-limsi-slu	21.5%	1442	0.4391	1421	0.4406
shef-cru-cuhtk-s1plu	26.6%	1455	0.4311	1441	0.4364
shef-b2u	26.7%	1386	0.4299	1374	0.4333
shef-b1u	27.5%	1393	0.4301	1375	0.4311
shef-s2u	29.2%	1418	0.4351	1401	0.4451
shef-s1u	32.0%	1393	0.4247	1387	0.4351

Table 5: Comparison of document rescoring methods for the SBU task. DERB refers to the equation presented in Section 4.2.3. MAX refers simply to the rescoring of a recombined document by giving it the highest score of all the individual documents.

matic segmentation is due to the mismatch between the real story boundaries and those defined automatically.

5. CONCLUSIONS

1. The TREC-8 SDR track evaluated current SDR technology on a substantial corpus of broadcast news material. The results show that information retrieval performance was not significantly affected by the size of the corpus or the increased number of out of vocabulary words caused by the language model becoming out of date. In general, problems caused by transcription errors are largely offset by techniques such as query expansion.
2. Automatic segmentation of the corpus with a very simple algorithm resulted in a 10% absolute degradation in average precision. Although this is a considerable loss of performance, the information retrieval capability of the system is still quite respectable with, on average, over 50% of the top 10 documents retrieved being relevant.

6. REFERENCES

- [1] J. Garofolo, E. Voorhees, C. Auzanne, and V. Stanford, "Spoken document retrieval: 1998 evaluation and investigation of new metrics," in *Proc. ESCA ETRW Workshop Accessing Information in Spoken Audio*, (Cambridge), pp. 1-7, 1999.
- [2] A. J. Robinson, G. D. Cook, D. P. W. Ellis, E. Fosler-Lussier, S. J. Renals, and D. A. G. Williams, "Connectionist Speech Recognition of Broadcast News." Submitted to *Speech Communication*.
- [3] S. Renals, D. Abberley, G. Cook, and T. Robinson, "THISL spoken document retrieval at TREC-7," in *Proc. Seventh Text Retrieval Conference (TREC-7)*, 1999.
- [4] H. Bourlard and N. Morgan, *Connectionist Speech Recognition—A Hybrid Approach*. Kluwer Academic, 1994.
- [5] A. J. Robinson, "The application of recurrent nets to phone probability estimation," *IEEE Trans. Neural Networks*, vol. 5, pp. 298-305, 1994.
- [6] S. Renals and M. Hochberg, "Start-synchronous search for large vocabulary continuous speech recognition," *IEEE Trans. Speech and Audio Processing*, vol. 7, pp. 542-553, 1999.

SBU Run	Retrieved	AveP
shef-slu	1393	0.4247
shef-s1 (no adverts)	1415	0.4376

Table 6: Story Boundary Unknown condition: effect of indexing non-news material.

- [7] G. Williams and S. Renals, "Confidence measures from local posterior probability estimates," *Computer Speech and Language*, vol. 13, pp. 395–411, 1999.
- [8] T. Robinson and J. Christie, "Time-first search for large vocabulary speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Seattle), pp. 829–832, 1998.
- [9] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Amer.*, vol. 87, pp. 1738–1752, 1990.
- [10] B. Kingsbury, *Perceptually-inspired Signal Processing Strategies for Robust Speech Recognition in Reverberant Environments*. PhD thesis, Dept. of EECS, UC Berkeley, 1998.
- [11] B. E. D. Kingsbury, N. Morgan, and S. Greenberg, "Robust speech recognition using the modulation spectrogram," *Speech Communication*, vol. 25, pp. 117–132, 1998.
- [12] P. Clarkson and R. Rosenfeld, "Statistical language modeling using the CMU-Cambridge toolkit," in *Eurospeech-97*, pp. 2707–2710, 1997.
- [13] S. E. Robertson and K. Sparck Jones, "Simple proven approaches to text retrieval," Tech. Rep. TR356, Cambridge University Computer Laboratory, 1997.
- [14] J. Allan, J. Callan, W. B. Croft, L. Ballesteros, D. Byrd, R. Swan, and J. Xu, "INQUERY does battle with TREC-6," in *Proc. Sixth Text Retrieval Conference (TREC-6)*, pp. 169–206, 1998.
- [15] J. Xu and W. B. Croft, "Query expansion using local and global document analysis," in *Proc. ACM SIGIR*, 1996.
- [16] D. K. Harman, "Relevance feedback and other query modification techniques," in *Information Retrieval: Data Structures and Algorithms* (W. B. Frakes and R. Baeza-Yates, eds.), ch. 11, pp. 241–263, Prentice Hall, 1992.
- [17] G. Williams and D. Ellis, "Speech/music discrimination based on posterior probability features," in *Eurospeech-99*, 1999.
- [18] S. E. Johnson, P. Jourlin, K. S. Jones, and P. C. Woodland, "Spoken document retrieval for trec-8 at cambridge university," in *Proc. TREC-8*, 1999.
- [19] D. Abberley, D. Kirby, S. Renals, and T. Robinson, "The THISL broadcast news retrieval system," in *Proc. ESCA ETRW Workshop Accessing Information in Spoken Audio*, (Cambridge), pp. 14–19, 1999.
- [20] A. F. Smeaton, M. Morony, G. Quinn, and R. Scaife, "Taiscéalái: Information retrieval from an archive of spoken radio news," in *Proc. Second European Digital Libraries Conference*, 1998.
- [21] G. Quinn and A. Smeaton, "Optimal parameters for segmenting a stream of audio into speech documents," in *Proc. ESCA ETRW Workshop Accessing Information in Spoken Audio*, (Cambridge), pp. 96–101, 1999.
- [22] T. Robinson, D. Abberley, D. Kirby, and S. Renals, "Recognition, indexing and retrieval of british broadcast news with the thisl system," in *Eurospeech-99*, 1999.

University of Sheffield TREC-8 Q & A System

Kevin Humphreys^a Robert Gaizauskas^a
Mark Hepple^a Mark Sanderson^b
{k.humphreys,r.gaizauskas,m.hepple}@dcs.shef.ac.uk
m.sanderson@shef.ac.uk

^aDepartment of Computer Science / ^bDepartment of Information Studies
University of Sheffield
Regent Court, Portobello Road
Sheffield S1 4DP UK

1 Introduction

The system entered by the University of Sheffield in the question answering track of TREC-8 is the result of coupling two existing technologies – information retrieval (IR) and information extraction (IE). In essence the approach is this: the IR system treats the question as a query and returns a set of top ranked documents or passages; the IE system uses NLP techniques to parse the question, analyse the top ranked documents or passages returned by the IR system, and instantiate a query variable in the semantic representation of the question against the semantic representation of the analysed documents or passages. Thus, while the IE system by no means attempts “full text understanding”, this approach is a relatively deep approach which attempts to work with meaning representations.

Since the information retrieval systems we used were not our own (AT&T and UMass) and were used more or less “off the shelf”, this paper concentrates on describing the modifications made to our existing information extraction system to allow it to participate in the Q & A task.

2 System Description

2.1 Overview

The key features of the system setup are shown in Figure 1. Firstly, the TREC document collection and each question were passed to two IR systems which treated the question as a query and returned top ranked documents or passages from the collection. As one IR system we used the AT&T supplied top docu-

ments which were made available to all participants by NIST; as the second we used the passage retrieval facilities of the University of Massachusetts Inquiry system [2] to return top ranked passages. Following this, for each question, the question itself and the top ranked documents or passages were processed by a slightly modified version of the LaSIE information extraction system [7], which we refer to below as QA-LaSIE. This yielded two sets of results which were entered separately for the evaluation – one corresponding to each of the IR systems used to filter the initial document collection.

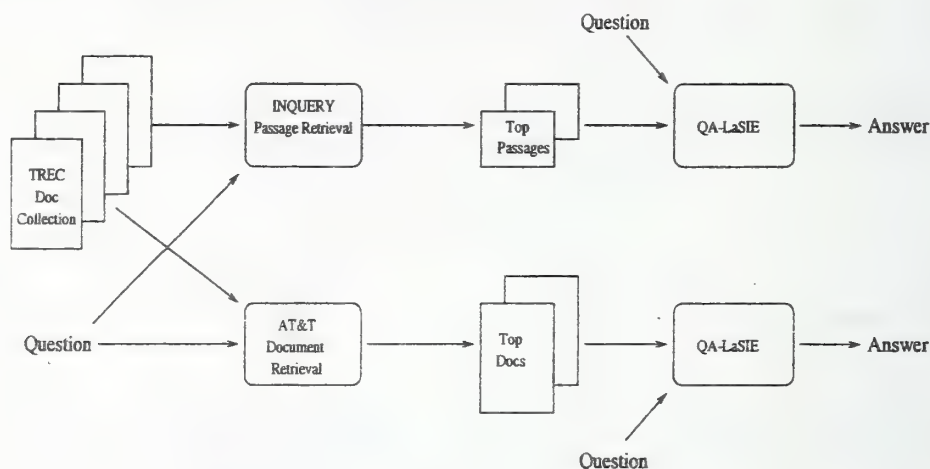


Figure 1: System Setup for the Q & A Task

The reasoning behind this choice of architecture is straightforward. The IE system can perform detailed linguistic analysis, but is quite slow and could not process the entire TREC collection for each query, or even realistically preprocess it in advance to allow for reasonable question answering performance during the test run. IR systems on the other hand are designed to process huge amounts of data. Thus, the hope was that by using an IR system as a filter to an IE system we could benefit from the strengths of each [6].

In the next section we describe the basic LaSIE system and then in succeeding sections proceed to describe the modifications made to it for the TREC-8 Q & A task.

2.2 LaSIE

The LaSIE system used to perform the detailed question and text analysis is largely unchanged from the IE system as entered in the most recent Message Understanding Conference evaluation (MUC-7) evaluation [7]. The principal components of the system are shown in Figure 2 as executed interactively through the GATE Graphical Interface [4]. The system is essentially a pipeline of modules each of which processes the entire text before the next is invoked. The following is a brief description of each of the component modules in the system:

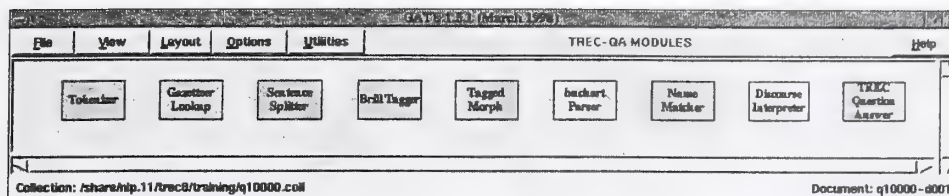


Figure 2: QA-LaSIE System Modules

Tokenizer Identifies token boundaries (as byte offsets into the text) and text section boundaries (text header, text body and any sections to be excluded from processing).

Gazetteer Lookup Identifies single and multi-word matches against multiple domain specific full name (locations, organisations, etc.) and keyword (company designators, person first names, etc.) lists, and tags matching phrases with appropriate name categories.

Sentence Splitter Identifies sentence boundaries in the text body.

Brill Tagger [1] Assigns one of the 48 Penn TreeBank part-of-speech tags to each token in the text.

Tagged Morph Simple morphological analysis to identify the root form and inflectional suffix for tokens which have been tagged as noun or verb.

Parser Performs two pass bottom-up chart parsing, pass one with a special named entity grammar, and pass two with a general phrasal grammar. A 'best parse' is then selected, which may be only a partial parse, and a predicate-argument representation, or quasi-logical form (QLF), of each sentence is constructed compositionally.

Name Matcher Matches variants of named entities across the text.

Discourse Interpreter Adds the QLF representation to a semantic net, which encodes the system's world and domain knowledge as a hierarchy of concepts. Additional information inferred from the input is also added to the model, and coreference resolution is attempted between instances mentioned in the text, producing an updated discourse model. A representation of the question is then matched against the model, using the coreference mechanism.

Question Answer Selects the required answer text using the resolved question representation in the discourse model.

2.3 QA-LaSIE

The QA-LaSIE system operates by processing an ordered set of texts for each question with the question itself as the first text. The IR systems' results were split into a subdirectory for each question, containing, firstly, the question itself, then, in rank order, a predefined number of texts or passages retrieved for that

question. For the Inquiry data, the top 10 passages were used, and for the AT&T data, the top 5 full texts. These limits were chosen mainly to restrict the system's total processing time, but for the Inquiry data the limit was based on a partial analysis of the rankings of texts containing a correct answer for the training set of questions.

For the evaluation, QA-LaSIE was run in batch mode to process each subdirectory of question plus retrieved texts. When an answer was found, 50- and 250-byte responses were written out, and processing moved immediately to the next question, as described below. The system required an average of around 15 minutes to process each question and its corresponding set of retrieved texts on a SUN Sparc 5 machine, though no effort has been spent on optimisation.

The following subsections detail the modifications required for the original IE system to operate in a question answering mode.

2.3.1 Question Parsing

An additional subgrammar was added to the phrasal parsing stage for interrogative constructions, which were not handled at all in the original LaSIE system. The grammar was developed until reasonable coverage on the 37 questions in the training set was obtained, with only a very limited attempt to cover constructions outside this set. Compositional semantic rules on each syntactic rule are used to build up a 'quasi-logical form' (QLF) representation, in the same way as the rest of the grammar. A special semantic predicate, *qvar* (question variable), is used in the semantics to indicate the 'entity' requested by the question. For example, the question *Who composed Eugene Onegin?* would produce the following QLF representation:

```
qvar(e1), person(e1)
name(e2, 'Eugene Onegin')
compose(e3), tense(e3, past)
lsubj(e3, e1), lobj(e3, e2)
```

Here, each entity in the question gives rise to a unique identifier of the form *en*. The use of the lexical item *who* causes the addition of *person(e1)*, but the semantic class of *e2* (*Eugene Onegin*) is unspecified. The relational predicates *lsubj* (logical subject) and *lobj* (logical object) simply link any verb arguments found in the text, rather than using any subcategorisation information to determine the arguments required for a particular verb.

The QLF representation of each question is stored for use in the subsequent processing of each candidate answer text. After parsing, the question is processed by the Namematcher and Discourse Interpreter modules, but the results of these modules are currently unused. Potentially, these modules could carry out coreference resolution within the question, thus allowing complex, even multi-sentence, questions to be processed, but this capability was not required for any of the questions in the training set and was not used for the test run.

2.3.2 Question Resolution

The candidate texts for each question are processed exactly as in the standard LaSIE system, up until the completion of the Discourse Interpreter stage. At this point, if a stored representation of a question for the current text is found, this representation is examined and an attempt made to find an answer within the text's completed discourse model. Each question representation gives rise to a hypothesised entity (the *qvar*), and the Discourse Interpreter's general coreference mechanism is used to attempt to find an 'antecedent' for the hypothesis from the text.

Various restrictions are placed on the hypothesised entity from the question's QLF representation. The entity required to answer the question will be flagged as having the semantic class *qvar*, but it may also have other semantic types, such as *person* if the question introduces the entity using *Who*, as in the example above. The entity may also have other attributes mentioned in the question, such as *name*, and attributes linking the *qvar* entity to other entities from the question, in particular the verb argument relations *lsubj* and *lobj*.

In some cases the question grammar may fail to parse a question as an interrogative construction, and the parser will produce only a partial QLF representation which does not include a *qvar*. In this case the discourse interpreter applies a fallback mechanism to force the first text in each question/answer set to be interpreted as a question, simply treating the first entity in a QLF representation with no *qvar* as the *qvar*. The first entity is currently chosen arbitrarily, with no analysis of the partial QLF representation, but the mechanism does allow the system to recover from the incomplete coverage of the question grammar, and still produce answers even where no question was recognised.

Anaphor Resolution Before attempting to resolve the *qvar* entity, the general coreference mechanism is applied to any other entities from the question. The coreference mechanism currently only attempts to resolve the classes of anaphora defined for the MUC-7 evaluation, i.e. identity relations between proper names, pronouns, noun phrase heads and noun modifiers. No general attempt is currently made to resolve multiple descriptions of events in a text, though this is attempted for question resolution, as described below.

The general coreference mechanism, described fully in [5], acts to compare pairs of entities to determine a similarity measure. Firstly, the semantic classes of the two entities are compared (semantic type compatibility) by testing for a dominance relation within the system's ontology, or concept hierarchy. Secondly, if the semantic classes are compatible, the values of all 'immutable' (fixed single-valued) attributes (e.g. *gender*, *number*) are compared (attribute similarity) to ensure no conflicts exist. Thirdly, an overall similarity score is calculated, combining the distance between the semantic classes of the two instances, and the number of shared, non-immutable attributes.

For each potential anaphor, if any comparison pairs are assigned a similarity score, the entity with the highest score will be merged with the anaphor in the discourse model. This results in the representation of a single entity in the

discourse model which has multiple realisations in the text, i.e. a coreferential entity.

Event Similarity For hypothesised *qvar* question answer entities, an additional, fourth, comparison stage has been added to the coreference mechanism to ensure that a candidate antecedent, or answer, shares any relations to *event* entities (*lsubj*, *lobj* or *comp* (complement)). This is required to allow the resolution of the *qvar* from a question like *Who composed Eugene Onegin* with an entity from a text containing *Tchaikovsky wrote Eugene Onegin*. The *qvar* entity here is the logical subject of the *compose* event, but to resolve this with *Tchaikovsky*, the candidate antecedent must have a *lsubj* relation with an event of a compatible class and with the same arguments, *lobj* in this case, via coreference between the question and the text.

This additional stage therefore requires the identification of events of compatible classes, testing semantic type similarity within the system's ontology. However, rather than explicitly extending the ontology to include as many concepts as possible, and introducing all the problems of word sense ambiguity, a simple high-level general ontology was defined, and then reference made to WordNet [3] hypernym/hyponym relations during processing. When attempting to find an antecedent for the *qvar* above, the *compose* event would be compared with the *write* event using the relations between WordNet synsets. An arbitrary limit of 3 hypernym/hyponym links was used to constrain the event similarity test, and, in this case, only a single link is required in WordNet to relate *compose* and *write*. The distance between the two event classes is then combined with the general coreference mechanism's similarity score for the *qvar* antecedent, so preferring antecedents which are arguments of more similar event classes.

The copular verb *be* was treated specially when comparing it to other event classes. The grammar treats the copular as any other verb, introducing an event instance for it, but in the event similarity test it is treated as being compatible with any other event class, though with a low score.

The general approach to ontology construction in the LaSIE system has previously been to only include concepts directly relevant to a particular IE task. The tasks have been fixed and well defined, so a small domain-specific ontology has been sufficient. For the Q & A task, however, no assumptions about the domain of each question can be made, and so a more general purpose ontology is required. Reference to the WordNet hierarchy is currently only made for comparing event classes. A similar comparison could also be made for object classes, effectively extending the system's object hierarchy as necessary, but this was not implemented for the Q & A evaluation.

2.3.3 Answer Generation

An additional Q & A task-specific module was added to the LaSIE system, following the Discourse Interpreter stage. This module simply scanned the final discourse model for each text to check for an instantiated *qvar*, i.e. a *qvar* that had been successfully resolved with an entity in the text. If found, the realisation

of that entity in the text (the longest in the case of multiple realisations via coreference resolution) was used as the central point from which 50- and 250-byte text windows were extracted to be used as question responses.

A significant feature of the QA-LaSIE system's operation is that once a response for a particular question has been produced, no further candidate texts are processed for that question. This was partly to improve system performance by avoiding any unnecessary processing of texts once an answer had been produced. However, this did assume that the IR systems' ranking of the candidate texts was accurate. The highest ranked text was processed first, and if an answer was produced from it, lower ranked texts were not considered. With hindsight, this approach was really at odds with the Q & A task's intended mode of operation, where multiple ranked answers for each question were expected. The QA-LaSIE system could easily be adapted to return multiple answers, and re-use the IR systems' rankings, but the single-answer mode reflects the original IE approach.

3 Results and Analysis

Since the QA-LaSIE system only ever produced a single answer for each question, which was arbitrarily assigned a ranking of 1, the official results evaluating the accuracy of system rankings are not particularly meaningful for QA-LaSIE¹. Therefore, an initial analysis of the system results has been carried out to attempt to express performance in the standard recall and precision metrics (in this context recall is the proportion of questions correctly answered, precision the proportion of answered questions for which the answer is correct).

The following results were obtained from the individual judgements of question answers and an analysis of the system's intermediate outputs for each question.

For the NIST-supplied AT&T data, where the top 5 full texts for each question were processed, the overall results were:

50-byte answers:	250-byte answers:
Recall = 14 / 198 = 7.07%	Recall = 19 / 198 = 9.59%
Precision = 14 / 60 = 23.33%	Precision = 19 / 60 = 31.67%

For the University of Massachusetts Inquiry data, where the top 10 passages for each question were processed, the overall results were:

50-byte answers:	250-byte answers:
Recall = 16 / 198 = 8.08%	Recall = 22 / 198 = 11.11%
Precision = 16 / 61 = 26.23%	Precision = 22 / 61 = 36.06%

A more detailed analysis of the QA-LaSIE results alone, separate from the retrieval system, was then carried out. This involved attempting to determine,

¹The adjudicated mean reciprocal rank scores were as follows. For 250-byte answers, .111 for the Inquiry supplied top 10 passages, .096 for the AT&T supplied top 5 full texts. For 50-byte answers, .081 for the Inquiry data, .071 for the AT&T supplied data.

for each question, whether the retrieval results used did in fact include a text containing an answer. To avoid manually judging every text, the Q & A task judgements of all system results were used. The definition of a correctly retrieved text is therefore a text from which any system in the evaluation produced a correctly judged answer, though clearly there may be other retrieved texts which also contain answers. Using this definition, the top 5 texts from the AT&T data represented 71.72% recall of correct question answers, and the top 10 passages from the Inquiry data represented 76.26% recall (though no manual test has been done to ensure the correct passages were selected from the texts).

Analysing the QA-LaSIE results for only those questions for which texts were correctly retrieved produced the following figures for the AT&T data:

50-byte answers:	250-byte answers:
Recall = 14 / 141 = 9.87%	Recall = 19 / 141 = 13.38%
Precision = 14 / 47 = 29.79%	Precision = 19 / 47 = 40.43%

and for the Inquiry data:

50-byte answers:	250-byte answers:
Recall = 16 / 151 = 10.60%	Recall = 22 / 151 = 14.57%
Precision = 16 / 49 = 32.65%	Precision = 22 / 49 = 44.90%

A further analysis considered system performance for only those questions which were parsed as interrogative constructions (i.e. where the QLF representation included a `qvar`), and where texts containing an answer were correctly retrieved. This excludes some cases where the system produced answers, some correct, despite the QLF representation of the question containing no `qvar`, using the fallback mechanism described in Section 2.3.2. For the AT&T data, the results are:

50-byte answers:	250-byte answers:
Recall = 13 / 87 = 14.94%	Recall = 17 / 87 = 19.54%
Precision = 13 / 42 = 30.95%	Precision = 17 / 42 = 40.48%

and for the Inquiry data:

50-byte answers:	250-byte answers:
Recall = 12 / 84 = 14.28%	Recall = 18 / 84 = 21.43%
Precision = 12 / 40 = 30.00%	Precision = 18 / 40 = 45.00%

These results give a better indication of the performance of the QA-LaSIE system alone, attempting to exclude the particular IR system used, and also the current incomplete state of the question grammar.

4 Conclusion

We have not yet carried out detailed failure analysis of the QA-LaSIE system, and so cannot make many specific claims about where the strengths and weaknesses of the approach lie. The performance of the system clearly leaves much

to be desired, and the results are very low using the reciprocal ranking measure used in the evaluation. However this measure, and indeed the current Q & A task methodology, does not allow any useful measure of precision. Answers to all test questions are known to be available in the test corpus, and the ability to return negative results where appropriate is not evaluated.

Given the very limited effort that went into tuning the QA-LaSIE system we believe that the approach performed sufficiently well to warrant further investigation. The system was assembled in less than two person weeks, and very little effort was available to adapt the general coreference mechanism to the task of question resolution.

Several areas where further work or investigation are clearly needed are:

- *Question Parsing* As only 2/3 of the questions were parsed more effort is needed to refine and extend the coverage of the question grammar.
- *Answer Text Processing* Analysis needs to be carried out to see to what extent the meaning representations computed for the answer texts do or do not contain the information required to answer the questions. If not, the source of this inadequacy needs to be identified (faulty parsing, inadequate lexical or world knowledge).
- *QVAR Coreference* Analysis of whether the *qvar* matching in the coreference mechanism is too weak or too strong needs to be carried out. Strict insistence that all attributes associated with the *qvar* in the question be matched in a candidate answer text may be too strong a requirement; on the other hand loosening the match may result in spurious answers.
- *General Purpose Ontology* The ontology used in the QA-LaSIE system, while intended to be general purpose, is actually abstracted from a small number of business domains used in the development of the LaSIE IE system. This clearly has only a very limited coverage of the varied domains represented in an unconstrained set of questions. Considerable further investigation into ways of extending the coverage is required, including evaluation of the use of available resources such as WordNet, as implemented here for event classes.
- *Multiple Answers* As noted, QA-LaSIE halts after returning the first answer it finds for each question. It would be relatively trivial to extend the system to process all the documents passed to it by the IR system and rank the resulting answers. The impact of this on performance needs to be assessed.

Such investigations will help to reveal whether the approach we have followed for the Q & A task is appropriate. More generally they will shed light on the very interesting questions this task throws up: to what extent are 'deeper' models of language processing necessary to perform a question answering task against large text collections.

Acknowledgements

The authors would like to thank James Allan and Daniella Malin of the Computer Science Department, University of Massachusetts for supplying the results of running the Inquiry system with the Q & A questions as queries against the TREC data collection.

References

- [1] E. Brill. A simple rule-based part-of-speech tagger. In *Proceeding of the Third Conference on Applied Natural Language Processing*, pages 152–155, Trento, Italy, 1992.
- [2] J.P. Callan, W.B. Croft, and S.M. Harding. The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert System Applications*, pages 78–83, 1992.
- [3] Miller G.A., R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Introduction to WordNet: On-line. Distributed with the WordNet Software., 1993.
- [4] R. Gaizauskas, H. Cunningham, Y. Wilks, P. Rodgers, and K. Humphreys. GATE – an Environment to Support Research and Development in Natural Language Engineering. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-96)*, pages 58–66, Toulouse, France, October 1996.
- [5] R. Gaizauskas and K. Humphreys. Quantitative Evaluation of Coreference Algorithms in an Information Extraction System. In S. Botley and T. McEnery, editors, *Discourse Anaphora and Anaphor Resolution*. Forthcoming. Also available as Department of Computer Science, University of Sheffield, Research Memorandum CS – 97 – 19, <http://www.dcs.shef.ac.uk/research/resmems>.
- [6] R. Gaizauskas and A.M. Robertson. Coupling information retrieval and information extraction: A new text technology for gathering information from the web. In *Proceedings of the 5th Computed-Assisted Information Searching on Internet Conference (RIAO'97)*, pages 356–370, Montreal, 1997.
- [7] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, and Y. Wilks. Description of the LaSIE-II system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

University of Surrey Participation in TREC 8:

Weirdness Indexing for Logical Document Extrapolation and Retrieval (WILDER)

Khurshid Ahmad
Lee Gillam
Lena Tostevin

AI Group
Department of Computing
School of EE, IT and Maths
University of Surrey
UK, GU2 5XH
K.Ahmad@surrey.ac.uk

Abstract

This paper describes the development of a prototype document retrieval system based on frequency calculations and corpora comparison techniques. The prototype, WILDER, generated simple frequency information based on which calculations of document relevance could be made. The prototype was built to allow the University of Surrey to debut in the U.S. Text Retrieval Competition (TREC).

User queries as specified by the TREC organisers were converted into simple word-frequency lists and compared against values for the entire corpus. These relative frequency values indicatively produced document relevance. The application of morphological and empirical heuristics enabled WILDER to produce the ranked frequency lists required.

Introduction

The *ad hoc* task of TREC8 investigates the performance of systems in ranking a static set of documents against novel topics (queries). For each topic, the top 1000 documents satisfying the topic are submitted. Recall and precision techniques are used on these rankings to determine the results of the competition overall.

We have used term identification and extraction techniques for identifying topics discussed in a given text. In this note we focus on the use of single word terms for identifying topics. The techniques are based on differences between general language texts, texts used in an everyday context, and special language texts. The special language texts are texts written, for instance, by scientists, engineers, business persons and hobbyists in their respective languages of physics, chemistry, engineering, business, and hobbies. English-speaking physicists will use the English rendering of terms of physics and use their knowledge of English language, which they share with other speakers of English. Similarly a Chinese speaking physicist writing in Chinese will use the Chinese rendering of terms plus their knowledge of Chinese which they share with other Chinese speakers. The special language texts can be distinguished from a collection of general language texts at different linguistic levels including lexical, morphological, syntactic and semantic. These differences can be measured quantitatively and qualitatively. Quantitative measures at the lexical level include frequency of usage of single and compound terms in special language texts and their equivalents in general language texts. Morphological differences can also be measured quantitatively by looking at the differences in the inflectional and derivational variants of terms; specialist texts comprise a larger number of plurals than used in general language; specialists use nominalised verbs more extensively than in general language.

The key difference at the lexical level, between specialist and general language texts, is in the distribution of the so-called open class words, typically nouns and adjectives, and the closed class words, typically determiners, conjunctions, prepositions and modal verbs. Consider the 100 million-word British

National Corpus (BNC) which 'was designed to characterise the state of contemporary British English in its various social and generic uses' (Aston and Burnard 1998); we will use the BNC as a general language corpus. The TREC 8 corpus in comparison to the BNC corpus can be regarded as specialist text corpus in that the former comprises financial and political news texts: about 30% of the text in TREC, measured by the number of documents, is derived from the London *Financial Times* and the other 45% is based on the *Federal Register* and *FBIS*. The potential general language component of TREC is based largely on the other 25% of the texts that are obtained from the *Los Angeles Times*. Tables 1a and 1b show the similarities and differences between the BNC and TREC-8 corpora in terms of the distribution of the 100 most frequently occurring tokens in the two. Note that the closed class words like determiners, prepositions and conjunctions have approximately the same distribution. The differences are in the number and appearance of the open class words. TREC-8 has 13 open class words, whereas the BNC can muster only 2. In the BNC, the first open class word *time* is the 79th most frequently word in the corpus, whereas in the TREC corpus the first open class word is *year* which is the 48th most used word in the corpus.

Table 1a. Distribution of 100 most frequent tokens in the British National Corpus (BNC comprises 4124 texts with over 100 Million tokens largely written and spoken during the 1970's and 1980's)

Tokens organised in order of frequency in batches of 10 at a time	Cumulative Relative Frequency	Number of Open Class Words
the, of, and, a, in, to, it, is, was, to	21.28%	0
i, for, you, he, be, with, on, that, by, at	6.66%	0
are, not, this, but, 's, they, his, from, had, she	4.35%	0
which, or, we, an, n't, 's, were, that, been, have	3.25%	0
their, has, would, what, will, there, if, can, all, her	2.42%	0
as, who, have, do, that, one, said, them, some, could	1.90%	0
him, into, its, then, two, when, up, time , my, out	1.57%	1
so, did, about, your, now, me, no, more, other, just	1.37%	0
these, also, people , any, first, only, new, may, very, should	1.18%	1
as, like, her, than, as, how, well, way, our, as	1.02%	0
Total Text (100106029 tokens)	45.01%	2

Table 1b. Distribution of 100 most frequent tokens in the TREC-8 Corpus (The corpus comprises 528155 texts with over 600 Million tokens largely written the 1990's)

Tokens organised in order of frequency in batches of 10 at a time	Cumulative Relative Frequency	Number of Open Class Words
the, of, to, and, in, a, for, that, is, s	22.36%	0
on, with, by, be, it, as, at, was, are, from	5.47%	0
this, said, will, has, not, have, he, an, or, which	3.76%	0
but, its, i, they, we, his, would, year , been, their	2.40%	1
were, who, one, had, more, mr , all, 1, new, per	1.88%	2
there, no, also, about, up, than, other, if, hyph, government	1.58%	1
two, cent , may, out, when, after, 2, last, state , 0	1.34%	2
first, pounds , people , only, can, you, time , some, over, company	1.21%	4
into, such, market , should, any, under, years , so, us, these	1.05%	2
what, t, 3, because, ft , 94, do, could, most, now	0.93%	1
Total Text (255637339 tokens)	41.98%	13

Table 1c shows the distribution of the open and closed class words in the various sub-corpora of the TREC -8 corpus. It appears that the *Federal Register* has the largest number of open class words amongst its first 100 words, followed by *FBIS*, and the *FT*. *LA Times* behaves differently in that it has only a 1/3rd of the open class words amongst its 100 most frequent words when compared to a similar number in the *Federal Register*. Recall that the BNC has only 2 open class words amongst the 100 most frequent words: A simple χ -square test will show that these subcorpora are different from the BNC on the basis of the frequency of open class words amongst the 100 most frequent words.

Table 1c. Distribution of open and closed class words in the TREC subcorpora

Group	<i>Federal Register</i>	<i>FBIS</i>	<i>Finacial Times</i>	<i>LosAngeles Times</i>	Row Total
(1)	(2)	(3)	(4)	(5)	(6)
Open Class	33	26	21	11	91 (22.7%)
Closed Class	67	74	79	89	309 (77.3%)
Column Total	100	100	100	100	400

It has been argued elsewhere that there are substantive differences at the morphological level in the use of keywords and certain verbs in the more formal literature of science and technology when compared to general language texts (see, for instance, Biber, Conrad and Reppen 1998). In the *FBIS* subcorpus the inflected forms *countries*, *elections*, and *relations*, and the derived forms *European*, *Russian*, and *Spanish* are respectively more frequent than *country*, *election*, *relation*, *Europe*, *Russia* and *Spain*. Similarly in the *FT* subcorpus *pounds*, *dollars* and *shares* are more frequent than their singular forms; and, there is little difference in the frequency of *company/companies* and *share/shares*. (In the *BNC* we note that *shares* are more frequent than *share*, but the term *dollar* is used 4 times more than the plural form). The *LA Times* subcorpus, however, does not have the same characteristics in that not only it has only 11 open class words amongst the 100 most frequent words, it has no plurals or nominalised verbs either amongst the 100 most frequently used words.

The lexical and morphological differences can help in filtering closed class words from special language texts and also certain commonly used open class words. This filtering process, should in principle, will result in a list of words that may be more closely related to the topic or theme of the paper. Some of the open class words or terms are usually *carriers* of meaning in that such words are used generally as a part of a complex phrase; for instance, the term *virus*, is used frequently in virology texts but occurs mostly as a part of a compound like *African Green Monkey virus* or *AIDS virus*. The meaning associated with the stem *virus* is related to the context of its usage in specialist texts. Similarly, the term *dollar* does not convey much information in international finance texts unless the context is examined, for example, whether the author of a given text was discussing *US \$*, *Australian \$* or *dollar-denominated bonds*. The following example illustrates the point made above. This is especially true if the specialist lexical item has entered general language vocabulary

We have carried out an experiment in which we removed the first 100 and then first 2000 most frequently occurring words in the *BNC* from the frequency lists compiled from the *FT*, *FBIS*, *LA Times* and the *FR* subcorpora. Tables 1d and 1e show the filtered wordlists from the *FT* subcorpus after the 100 and 2000 words from the *BNC* were excluded from the *FT* lists.

Table 1d. The residual, frequency ordered wordlist for the *FT* subcorpus after the first 100 most frequently words (occurring in the *BNC*) were removed.

mr,per,cent,pounds,year,ft,company,market,us,last	0.033063322
dollars,government,over,group,uk,yesterday,0,after,1,companies	0.016034449
bank,years,most,business,says,such,international,shares,world,2	0.010966193
however,news,tax,european,between,94,week,industry,93,share	0.009264547
three,interest,next,against,sales,profits,92,investment,while,london	0.008306062

Table 1e. The residual, frequency ordered wordlist for the FT subcorpus after the first 2000 most frequently words (occurring in the BNC) were removed.

cent,ft,dollars,0,94,93,92,markets,amp,investors	0.015253328
trading,cut,chief,index,finance,earnings,net,fall,turnover,japanese	0.004587571
according,losses,pre,announced,inflation,increased,non,debt,least,operating	0.0035461911
recovery,dividend,average,bond,spending,china,talks,recession,biggest,co	0.002904808
equity,currency,stake,official,analysts,trust,shareholders,assets,businesses,securities	0.002577078

Table 1d shows some of the keywords that form the basis of the English variant of the special language of finance and commerce. *FT* it appears focuses on *dollars*, *pounds*, *shares* and *industry*. Table 1e shows that when we remove the first 2000 most frequent words from *FT*'s wordlist we are dealing with more specific issues like *markets*, *investors*, *earning* and *losses*.

Weirdness of special language texts

The differences in the distribution of certain lexical items, and their variants, in special and general language texts can be quantified in terms of the relative frequencies of a specialist text (corpus) and a general language text corpus. We call this ratio an index of *weirdness* of a specialist text. This weirdness is used by an accentuated, and perhaps an eccentric, choice of lexical items measured in terms of their frequency of occurrence. Most weird words in a text will tend to represent it more closely than those that are not as weird. If the ratio is unity, then the lexical item has the same frequency in both general and special language; if the ratio is greater than unity then the item is used more frequently in specialist text then is the case for general language and vice versa. (The anthropologist Bronislaw Malinowski used the term *weird* to describe the language of *shamans* of South Sea Islands because they were using lots of names of spirits and objects).

It can be argued that comparison of the frequency distribution of items in special-language and general-language texts can identify signatures of a specialism. This technique has the advantage of being language-independent once the general-language corpus - or even a frequency list - has been obtained. 'Closed-class' words will tend to have ratios of around 1:1 in this comparison whereas terms or term carriers - content words rather than form words - will have a much higher ratio since their frequency in general-language texts will be low or potentially zero.

$$\text{Weirdness} = \frac{w_s / t_s}{w_g / t_g}$$

Where: w_s = frequency of word in specialist language corpus
 w_g = frequency of word in general language corpus
 t_s = total count of words in specialist language corpus
 t_g = total count of words in general language corpus

Consider the weirdness coefficients of some of the most frequent terms used in the *TREC-8* corpus; we have used BNC relative frequencies to compute the ratio.

	Freq (BNC)	Rel Freq (BNC)	Freq (TREC-8)	Rel Freq (TREC-8)	Weirdness
Dollar	2023	2.02086E-05	30450	0.000119114	5.894233822
Dollars	1677	1.67522E-05	182147	0.000712521	42.53289129
Government	62163	0.000620972	383115	0.001498666	2.413421274
Governments	4731	4.72599E-05	26413	0.000103322	2.186254565
Islam	523	5.22446E-06	4108	1.60696E-05	3.075846739
Islamic	1290	1.28863E-05	19410	7.59279E-05	5.892122549

Market	23719	0.000236939	278277	0.001088562	4.594273903
Markets	3895	3.89087E-05	80749	0.000315873	8.118310106

Recall the differences between the TREC corpus and the BNC. The BNC is a weighted corpus containing much diversity of general language texts so that no specific topic or domain has dominance. In the TREC corpus, governmental, financial and personal information are highly frequent, evident in the number of nouns occurring in the top 10 percentiles above. These 100 tokens make up 45% and 42% of the entire collection of the texts, representing 45,057,724 and 107,324,924 tokens respectively.

An immediate consequence of this fact is that analysis of the TREC corpus is considerably varied in contrast to that of the British National Corpus, biased towards these nouns. This information needs to be factored out of any contrastive analysis within the data.

Method

In order to compute the relevance of a given text to a query posed in TREC-8, the following steps shown in figure 7.1 were taken:

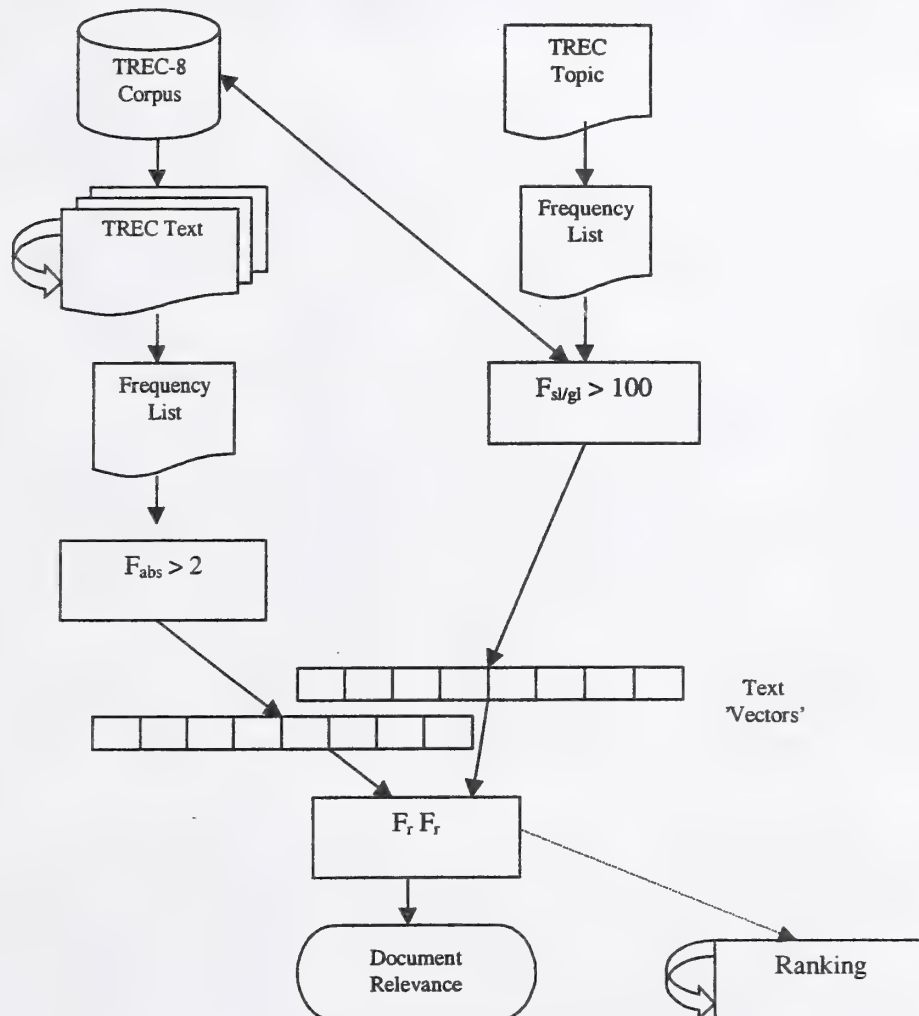


Figure 1: Steps to relevance

The resulting 'Vectors' - text and topic were then compared using the following correlation:

$$\sum fr_{sl} fr_{gl} - \sum fr_{sl}^m$$

Relevance was computed over the TREC-8 corpus for each topic and the texts were ranked. The 1000 most relevant texts were selected and submitted.

Consider Topic 444 in TREC-8:

<top>

<num> Number: 444

<title> supercritical fluids

<desc> Description:

What are the potential uses for supercritical fluids as an environmental protection measure?

<narr> Narrative:

To be relevant, a document must indicate that the fluid involved is achieved by a process of pressurization producing the supercritical fluid.

</top>

After removing words with Weirdness ≤ 100 we obtain the following weirdness-ordered wordlist:

WORD	FREQ	ABS FREQ	WEIRDNESS
achieved	1	0.0227	276.0000
document	1	0.0227	182.0000
fluid	4	0.0910	4780.0000
indicate	1	0.0227	590.0000
involved	1	0.0227	135.0000
measure	1	0.0227	249.0000
Potential	1	0.0227	133.0000
pressurization	1	0.0227	50500.0000
producing	1	0.0227	469.0000
protection	1	0.0227	122.0000
relevant	1	0.0227	370.0000
supercritical	3	0.0682	236000.0000
uses	1	0.0227	413.0000

TREC-8 determined the following texts to be relevant to this topic:

FBIS4-20472	FBIS4-44730
FBIS4-44741	FBIS4-44747
FBIS4-44913	FBIS4-45803
FBIS4-66450	FR940128-2-00102
FR940318-0-00170	FR940318-0-00172
FR940318-0-00173	FR940318-0-00213
FR940607-0-00051	FR940721-2-00028
FT932-7115	FT933-14063
FT943-4354	

Of these texts, in the first 10 that we selected, we now know that the texts marked in bold were relevant to this query:

```

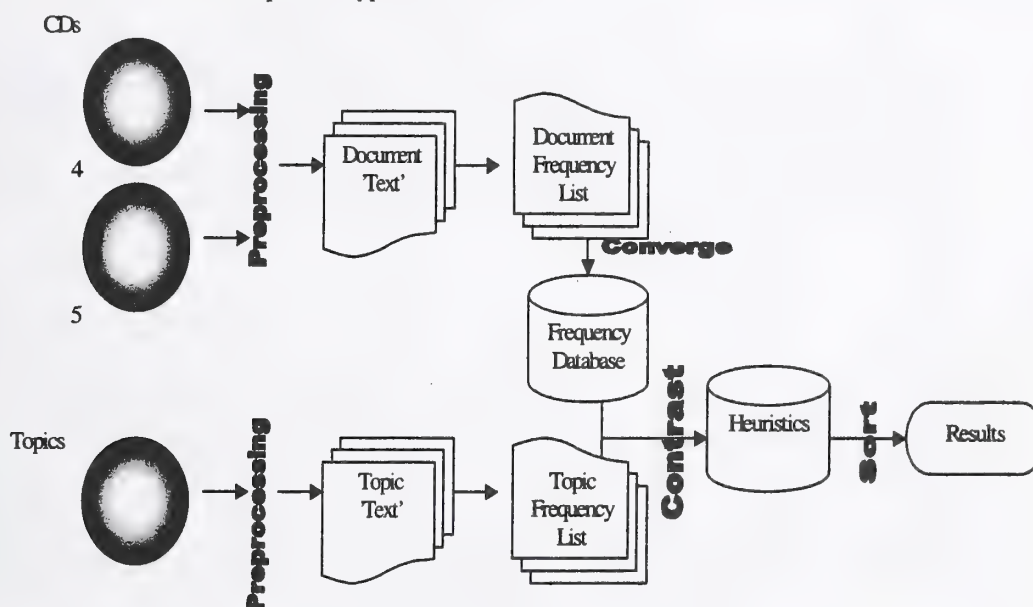
444 Q0      FBIS4-20472 0  0.000471 surfahi2
444 Q0      FBIS4-44913 1 -0.051765 surfahi2
444 Q0      FBIS4-44747 2 -0.207757 surfahi2
444 Q0      FBIS4-45803 3 -0.207972 surfahi2
444 Q0      FR940318-0-00173 4 -0.208627 surfahi2
444 Q0      FBIS3-41666 5 -0.209205 surfahi2
444 Q0      FR941206-1-00134 6 -0.209513 surfahi2
444 Q0      FBIS3-40501 7 -0.209580 surfahi2
444 Q0      FR940812-2-00056 8 -0.209600 surfahi2
444 Q0      FBIS3-40450 9 -0.209660 surfahi2

```

The WILDER program

In order to participate in this task, a prototype system, WILDER was developed. The system was built from a combination of existing Java, Perl and C code, and Unix shell scripting and associated utilities to achieve significant performance and ease of development.

This architecture for WILDER is shown below, which allows for a number of modular elements which can be developed in parallel and allows for a number of contrast algorithms to be switched in and out of the model in order to evaluate specific hypotheses:



Run stats

All processing was done within the Sun Solaris system. Building the original comparison resources took approximately 4 actual days on a single Sparc Ultra 1 - 140. Subsequently, each query took approximately 8 hours to satisfy the query from the raw results. There are many available optimisations to the algorithm used.

Future Direction

We have argued that fully automated extraction system can be created using simple contrastive frequency techniques for Information Retrieval in order to identify the topic of specific texts. The relative length of each text - at an average of 3.6K - is indicative of a lack of intra-text synonymy or term variants as would be true of lengthy narrative reports.

Treatment of simple morphology, acronyms, proper names and abbreviation needs further consideration within this particular arena, as does the application of techniques such as LSI and raw synonymy. Potentially, varying the values chosen for the application of the heuristics may make improvements to this simple methodology. Our goal was to build a system capable of handling such volumes of text within workable time. It is now our goal, based upon the results we have achieved, to improve and optimize this system using we have learnt through participation in this competition.

References

- Aston, Guy and Burnard, Lou. (1998). *The BNC Handbook: Exploring the British National Corpus*. Edinburgh: Edinburgh University Press.
- Biber, Douglas, Conrad, Susan and Reppen, Randi. (1998). *Corpus Linguistics: Investigating Language Structure and Use*. Cambridge: Cambridge University Press.
- Salton, G and McGill, M.J. (1983) *Introduction to Modern Information Retrieval*. McGraw-Hill

The Mirror DBMS at TREC-8

Arjen P. de Vries and Djoerd Hiemstra
{arjen,hiemstra}@ctit.utwente.nl
CTIT, University of Twente
The Netherlands

Abstract

The database group at University of Twente participates in TREC-8 using the Mirror DBMS, a prototype database system especially designed for multimedia and web retrieval. From a database perspective, the purpose has been to check whether we can get sufficient performance, and to prepare for the very large corpus track in which we plan to participate next year. From an IR perspective, the experiments have been designed to learn more about the effect of the global statistics on the ranking.

1 Introduction

The Mirror DBMS [dV99] combines content management and data management in a single system. The main advantage of such integration is the facility to combine IR with traditional data retrieval. Furthermore, IR researchers can experiment more easily with new retrieval models, using and combining various sources of information. This is an important benefit for advanced IR research; web retrieval, speech retrieval, and cross-language retrieval, each require the use of several representations of content, which is hard to handle in the traditional file-based approach, and becomes too slow in traditional database systems.

In the Mirror DBMS, the IR retrieval model is completely integrated in the database architecture, emphasizing efficient set-oriented query processing. The support for information retrieval in our system is presented in detail in [dV98] and [dVW99]. It supports other types of media as well, which has been demonstrated in the image retrieval system prototype described in [dVvDBA99]. The main goal of our participation in TREC is to test if our system can handle larger data sets without too many problems. Also, we wanted to find out the effect of global statistics on the ranking.

This paper is organized as follows. Sections 2 and 3 review the design of the Mirror DBMS and its support for IR, and discuss its use for TREC processing. Section 4 explains the experimental setup and interprets our results. Section 5 discusses our experience with using the Mirror DBMS for TREC, followed by conclusions.

2 Design

A complete overview and motivation of all aspects of the design of the Mirror DBMS is presented in [dV99]. Although following a traditional three-schema architecture, it uses different

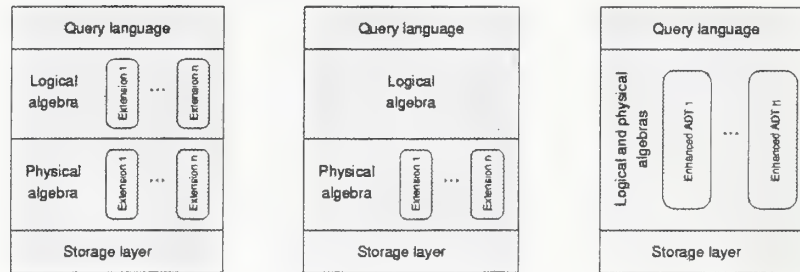


Figure 1: The multi-model DBMS architecture next to the extended relational and E-ADT DBMS architectures (from left to right).

data models at different levels: we therefore classify its design as **multi-model DBMS architecture**. The crucial architectural difference from other extensible database systems is that query processing at the logical layer uses *only* operators that are provided by the physical layer (see also Figure 1), and, domain-specific query processing (such as an IR extension) is defined *at the logical level primarily*. This choice enforces a system-wide physical data model and algebra spanning all extensions. Of course, the physical algebra can also be extended if necessary, i.e. when logical operations cannot be expressed efficiently in the physical algebra. The strict separation between the logical and physical levels allows using algebraic query optimization techniques, a key property of relational database management systems but hardly ever used in non-business application areas like content management.

The multi-model architecture provides the query processor with transparency through the layers. Put informally, query evaluation can ‘look down’ from the original request through all layers of the architecture. This should enable set-oriented query evaluation for almost every request, and allow maximal exploitation of parallelization and pipelining. In contrast, the black-box ADTs of ‘object-relational’ database systems restrict the DBMS in the possible manipulations of the query plans. This makes it more complicated to distribute and parallelize the query plans, or change the buffer strategy for iterative query processing as proposed in [JFS98]. Another alternative, the enhanced ADTs proposed by Seshadri [Ses98], provides little opportunity for optimizations that cross the boundaries between different extensions. Figure 1 compares these three architectures schematically.

3 Implementation

The prototype implementation of the Mirror DBMS uses Moa at the logical level, and Monet at the physical level. Monet is a parallel main-memory database system under development at the CWI in Amsterdam [BK95, BMK99], that is targeted as a backend system for various (query-intensive) application domains, such as GIS and data mining.¹ Moa is an object algebra studied in the database group at University of Twente, that is extensible with domain-specific structures. The Moa tools transform expressions in this algebra into sequences of operations in MIL, an algebra for the binary relational data model supported by Monet.

For the support of IR, we extended Moa with new structures at the logical level to handle document representation, ranking, and the computation of co-occurrence statistics. In com-

¹Monet is used successfully on a commercial basis by Data Distilleries, a start-up specializing in data mining applications.

document collection			intermediate results			
dj	ti	tfij	qdj	qti	qtfij	qntfij
1	a	2	1	a	2	0.796578
1	c	3	2	a	1	0.621442
2	a	1	2	b	2	0.900426
2	b	2				
2	e	2				

Table 1: Representation of content in BATs

bination with Moa's kernel support for collections and tuples, these structures can model a wide variety of IR retrieval models: the current prototype supports the well-known Okapi ranking scheme, InQuery's inference network retrieval model, as well as the linguistically motivated retrieval model (LMM, presented in Section 4.3). To illustrate, the following Moa expression ranks a collection of documents:

```
map[sum(THIS)](
  map[getBL(THIS, query, stats)]( docs )
);
```

The first `map` operation computes term probabilities for the query terms occurring in the document, using the global statistics specified in structure `stats`. The subsequent `map` combines these probabilities using a sum operation. Although this particular expression may not seem very interesting, the IR ranking operators can be combined with other operators such as `select`, resulting in a powerful query language.

The representation of the logical IR structures at the physical level is termed the **flattened representation** of the content. It consists of three binary tables (BATs), storing the frequency $tf(t_i, d_j)$ of term t_i in document d_j , for each term t_i occurring in document d_j . Table 1 illustrates this for a collection $\{d_1, d_2\}$ with documents $d_1 = [a, c, c, a, c]$ and $d_2 = [a, e, b, b, e]$. Computing the probability of relevance of the objects for query $q = [a, b]$ proceeds as follows. First, a table with the query terms is joined with the document terms in `ti` (the result is called `qti`). Next, (using additional joins) the document identifiers and the term frequencies are looked up (`qdj` and `qtfij`).² Finally, the retrieval status values are computed with some variant of the popular $tf \cdot idf$ ranking formula. To support these computations, Monet's physical algebra has to be extended with new operators, either in C or C++, or as a MIL procedure. The latter is preferable for easy experimentation; for example, the following MIL procedure computes the term probabilities given normalized term frequency and inverse document frequency using the LMM model:

```
PROC bel( nidfi, ntfig ) := {
  RETURN log( 1.0 + nidfi * ntfig * C );
}
```

An evaluation run processes 50 topics in batch, but the client interfaces of the Mirror DBMS

²Note that these joins are executed very efficiently, because the Moa structures make sure that the BATs remain *synchronized* all the time.

have been designed for interactive sessions with an end-user. Also, transferring the data from Monet to the Moa client has not been implemented optimally. Furthermore, optimizations such as using materialized views are not performed in the current Moa rewriter. These minor flaws would have inferred an unfair performance penalty to the evaluation of the architecture, and made logging the results rather cumbersome. Therefore, as a (temporary) solution, the MIL program generated by the Moa rewriter has been manually edited to loop over the 50 topics, log the computed ranking for each topic, and use two additional tables, one with precomputed normalized inverse document frequencies (a materialized view), and one with the document-specific constants for normalizing the term frequencies.

4 Experimental setup and results

Collection fusion is the process of merging the results of retrieval runs on separate, autonomous document collections into an effective combined result [VGJL95]. We have focused on this problem because large collections will be fragmented (horizontally) in several partitions, each managed by a separate server. Maintaining the exact global statistics induces an extra overhead, that may not be necessary if the fragments are sufficiently large.

Collection fusion is a trivial task for exact matching retrieval systems like systems using Boolean retrieval, but more complicated if a ranked retrieval system is used. In a number of publications on collection fusion it is argued that simply comparing similarity measures across subcollections leads to unsatisfactory results because of differences in the collection-dependent frequency counts [Bau97, CLC95, VF95, VGJL95]. One of the objectives of the TREC-8 evaluation described in this paper is to question this hypothesis. We feel that similarity measures across subcollections might in fact be comparable, but show worse evaluation results because of the evaluation setup.

4.1 Evaluation using the TREC collection

Relevance assessments on the TREC test collections are assembled by the pooling method: a pool of possibly relevant documents is created by taking the a sample of documents retrieved by each participating system. This pool is then shown to the human assessors [VH99a]. The sampling method used in TREC takes the top 100 of the retrieved documents of each participating system.

Since the start of TREC in 1992, the test collections have been used in numerous evaluations outside the official TREC. For these evaluations, all documents that were not in the top 100 of any of the official participating systems are assumed to be not relevant. But, evaluations that did not contribute to the TREC pool probably have unjudged documents in the top 100 making these evaluations less reliable than the official TREC evaluation. This is especially true for new, previously unexplored approaches to retrieval. If a systems finds relevant documents that no system was able to find before, then these documents will probably not be judged in an old TREC collection. *The only way to check the relevance of these documents is by official TREC participation.*

4.2 Conditions for naive collection fusion

Let us define 'naive' collection fusion as the process of merging the search results on the subcollections based on the document similarities. The first condition for naive collection fusion is that each subcollection uses the same retrieval model or weighting algorithm for retrieval. Secondly, we assume that each subcollection uses the same indexing vocabulary [Bau97]. A third condition is that subcollections are sufficiently large to allow for the reliable local estimation of document frequencies. If the subcollections are too small, ineffective retrieval on the subcollections will affect the merged result.

An evaluation of Callan et al. [CLC95] under these conditions for TREC topics 51-150 showed that naive merging was significantly worse than ranking based on globally estimated document frequencies, causing losses from 10-20% in average precision. But, the results of naive merging reported by Callan et al. [CLC95] were not part of an official TREC participation. It is likely that their merged run has a worse coverage of judgements, because the TREC-2 and 3 pools were (almost) only created by systems that use a central index for retrieval. Maybe, their merged run was as good as the central index run after all. To check this hypothesis, we decided to put up a retrieval run using naive merging for judging.

4.3 Some theoretical back-up for naive merging

The Mirror DBMS uses the linguistically motivated probabilistic model of information retrieval [Hie99, HK99]. The model builds a simple statistical language model for each document in the collection. The probability that a query T_1, T_2, \dots, T_n of length n is generated by the language model of the document with identifier D is defined by the following equation:

$$P(T_1=t_1, \dots, T_n=t_n|D=d) = \prod_{i=1}^n (\alpha_1 \frac{df(t_i)}{\sum_t df(t)} + \alpha_2 \frac{tf(t_i, d)}{\sum_t tf(t, d)}) \quad (1)$$

Equation 1 can be rewritten to a vector product formula by first dividing it by $\prod_{i=1}^n (\alpha_1 df(t_i) / \sum_t df(t))$ [Hie99]. This will not affect the ranking within a subcollection, but it will affect the final ranking after merging the search results of the separate subcollections, because we divided by collection specific document frequencies. It can be shown that the ranking of the vector product formula in table 2 approximates the ranking defined by the conditional probability $P(D|T_1, T_2, \dots, T_n)$ of a document being relevant given a query.

vector product formula:	$\text{similarity}(Q, D) = \sum_{k=1}^l w_{qk} \cdot w_{dk}$
query term weight:	$w_{qk} = tf(t_k, q)$
document term weight:	$w_{dk} = \log(1 + \frac{tf(t_k, d)}{df(t_k) \sum_t tf(t, d)}) \cdot \frac{\alpha_2 \sum_t df(t)}{\alpha_1}$

Table 2: $tf \cdot idf$ term weighting algorithm

From Bayes' rule we know that dividing equation 1 by $P(T_1, T_2, \dots, T_n)$ and multiplying it by $P(D)$ results in $P(D|T_1, T_2, \dots, T_n)$. For a large collection and a query that has a small number of hits, $tf(t, d)=0$ for most terms t and documents d . Therefore, $\prod_{i=1}^n (\alpha_1 df(t_i) / \sum_t df(t))$ approximates the marginal probability $P(T_1, T_2, \dots, T_n)$ and the ranking defined by table 2

approximates the ranking defined by $P(D|T_1, T_2, \dots, T_n)$. The a-priori probability $P(D=d)$ of a document d being relevant can be included by adding the logarithm of equation 2 to the similarities of table 2 as a final step.

$$P(D=d) = \frac{\sum_t tf(t,d)}{\sum_t \sum_d tf(t,d)} \quad (2)$$

We hypothesise that, if the approximation is not too far off, the result after merging is not significantly worse than what would have been possible with a central index.

4.4 Official results

Table 3 lists the official TREC runs. Global runs denote runs using the global collection statistics. Local runs denote the naive collection fusion runs, using local collection statistics on the four TREC subcollections: Federal Register, Foreign Broadcast Information Services, Los Angeles Times and Financial Times.

run name	description	avg. prec.
UT800	global run	0.260
UT803	global run; LCA	0.176
UT803b	global run; LCA from F.Times and LA Times	0.260
UT810	local run (judged)	0.043
UT813	local run; LCA from local	0.145

Table 3: official results

Unfortunately, our submitted official runs have been degraded by two bugs, that affected in particular the naive merging run that was judged by NIST. By our own mistake, the global runs have used the wrong (local) normalizing constant for the *idf*,³ an error in Monet's join implementation resulted in random answers for three of the four local runs. After fixing these bugs, the results of the global run UT800 improved from 0.260 to 0.275 and the results of the local run UT810 improved from 0.043 to 0.260. Table 4 lists the results on the four subcollections. Except for the Federal Register, which has hits for only 19 topics anyway, the average precision on the subcollections do not differ much at all. Unofficial runs, with these bugs fixed, are indicated in this paper by a 'u' postfix (so 'UT500u' is the fixed 'UT500' run).

run name	Fed.Reg.	FBIS	LATimes	F.Times	merged
UT800u (global)	0.326	0.317	0.279	0.356	0.275
UT810u (local)	0.351	0.319	0.276	0.356	0.260
topics w. hits	19	43	45	49	50

Table 4: average precision per subcollection after bug-fix

The merged local run is about 6% worse than the global run. This might be a significant difference according to some significance test, like e.g. the t-test [Hul93]; but, if so, it is still not valid to draw the conclusion that the global approach is indeed better than the naive merging approach. This conclusion would only be valid if both evaluations were done under identical, controlled, conditions; which they are not, because both runs were not judged by NIST and we do not control the other systems that contributed to the pool. Almost all systems that contributed to the TREC-8 pool were systems using the global approach.

³Strange enough, this mistake improves average precision slightly on the TREC-6 topics.

Therefore, the pool favours central index approaches over distributed index approaches if it is used to evaluate runs that did not contribute to the pool. This can be shown by looking at the percentage of documents that are judged for different cut-off levels of the fixed UT800u and UT810u runs. The percentage of documents in a run that are judged, will be called the judged fraction.

run name	P at 10	P at 30	P at 100	P at R	avg. P
UT800u (global)	0.496	0.378	0.234	0.319	0.275
UT810u (local)	0.436	0.343	0.222	0.310	0.260

run name	J at 10	J at 30	J at 100	J at R	
UT800u (global)	1.000	1.000	0.996	0.987	
UT810u (local)	0.984	0.978	0.952	0.947	

Table 5: merged results after bug-fix: a) precision; b) judged fraction

Table 5a and b show the precision and the judged fraction of the global and the local run at different cut-off levels. There is a major difference between the judged fractions of the global run and the local run. The global run misses 0.4% of the documents in its top 100. The local run misses 4.8% of the documents in the top 100, some of them are even missing in the top 10.

4.5 Local context analysis

Based on its success on InQuery at previous TREC conferences, we expected a significant improvement by using topics expanded with LCA [XC96]. Also, investigating the expansion terms, LCA seemed to do a good job. For example, on topic 311 (which is about industrial espionage), it finds terms like 'spy', 'intelligence', and 'counterintelligence', and from the financial times sub-collection it even identifies 'Opel', 'Volkswagen', and 'Lopez' as relevant terms. But, instead of improving the effectiveness of retrieval, the measured performance turned out to have degraded. Some tweaking of the parameters, reducing the weights of expansion terms and using fewer of them ($N=30$), the performance improved upon the baseline, but only slightly; on the runs submitted for TREC-8, it has degraded performance.

A possible explanation for these disappointing results is that the algorithm has been applied to documents instead of passages (as done in [XC96]), and the TREC collection itself was used to find expansion terms instead of another, larger collection. One result was that the varying length of documents had a large impact on the expansion terms chosen, which is undesirable. Another explanation is that LMM weighting provides such a high baseline, that it is very hard to improve upon. A comparison between the (impressive) baseline results of LMM on TREC-6 favours the latter explanation: because the performance of the Mirror DBMS with LMM weighting scheme, without LCA, was almost as good as InQuery's performance after using LCA. With the tweaked LCA, LMM weighting performed better on all reported precision and recall points, except for the precision at twenty retrieved documents, at which InQuery performed slightly better. On the TREC-8 topics it did not contribute positively to the results.

5 Discussion

Although more of anecdotal than scientific value, the story of our participation in TREC-8 with the Mirror DBMS illustrates the suitability of this architecture for experimental IR. Eight days before the deadline, it still seemed impossible to participate with this year's TREC, as Monet kept crashing while indexing the data; until, the seventh day, the new release suddenly made things work! We decided to try our luck and see how far we could get in a week; and we should admit, it has been a crazy week. It meant running the topics on TREC-6 first, to compare the results with the runs performed before; as well as changing the ranking formula to integrate document length normalization. In the weekend, we implemented the use of co-occurrence statistics (which has turned out to be not so useful as expected). So, in one week we managed to index the data, perform various experiments for calibration, run the best experiments on TREC-8, and submit five runs, just before the final deadline.

5.1 Efficiency

The machine on which the experiments have been performed is a Sun Ultra 4 with 1 Gb of main-memory, running SunOS 5.6. The machine is not a dedicated server, but shared with some other research groups as a 'compute server'. Monet effectively claims one processor completely while indexing the collection, or processing the fifty topics on each of the sub-collections. The division of the complete collection in five sub-collections (as it comes on different compact discs) is maintained. The topics are first run in each sub-collection, and the intermediate results are merged. Depending on the size of the sub-collection, estimating the top 1000 ranking takes between 20 seconds and two minutes per topic. How to further improve this execution performance is discussed below.

Preparation of the five sub-collections takes about six hours in total. Computing the table with document-specific term frequencies is performed using Monet's module for crosstables. But, using the grouping operation for all documents at once allocates all available memory, and eventually crashes the DBMS because it cannot get more, if it is run on the complete set of documents of any but the smallest sub-collection.⁴ Therefore, the indexing scripts run on fragments of the sub-collections at a time, and frequently write intermediate results to disk, obviously slowing down the process more than necessary.

5.2 The road ahead

The execution performance of the Mirror DBMS on TREC is clearly better than a naive (nested-loop) implementation in any imperative programming language, but, the obtained efficiency is not fast enough to beat the better stand-alone IR systems that also participate in TREC. But, compared to the techniques used in systems like InQuery (see [Bro95]), the current mapping between the logical and physical level is too straightforward: it does not use inverted files, has not fragmented the terms using their document frequency, and it ranks all documents even if only the beliefs for the top 1000 are used. Also, Monet should make it relatively easy to take advantage of parallelism in modern SMP workstations.

The merits of some possible improvements can only be evaluated experimentally. For ex-

⁴Notice that such problems are not necessarily solved by using commercial systems; Sarawagi et al. report similar memory problems with DB2 when using normal SQL queries for mining for associations hidden in large data sets [STA98].

ample, it is not so clear beforehand whether inverted files are really the way to go. Query processing with inverted files requires merging the inverted lists before beliefs can be computed, which is hard to perform without trashing the memory caches frequently; which has been shown a significant performance bottleneck on modern system architectures (see e.g. [BMK99] for experiments demonstrating this for Monet).

Without experiments, much improvement can be expected from fragmentation of the document representation BATs based on the document frequency, in combination with the 'unsafe' techniques for ranking reported in [Bro95]. Some preliminary experiments indicate a 100 times improvement with only a small loss in precision. Such (domain-specific) optimization techniques are easy to integrate in the mapping from Moa structures to MIL, thanks to the declarative nature of the algebraic approach. A similar argument applies to extending the Mirror DBMS with the buffer management techniques discussed in [JFS98]. In MIL, buffer management is equivalent to directing Monet to load and unload its tables. By integrating such directives in the generated MIL programs, it is expected that these improvements can also be added without many complications.

6 Conclusions

Without any additional algorithms, LMM ranking produces reasonably good results. Unfortunately, due to the bug in our experiments, we cannot yet give conclusive answers about the difference between using local or global statistics; but, we may conclude that the difference is rather small. Our current use of co-occurrence statistics has not improved our results, but further research is necessary in this area.

Despite of the flaws in the current implementation, we believe that the Mirror DBMS has proven to be a useful platform for IR experiments on the TREC data. The true benefits of its design will only be exploited when the system is developed further, and the indexing task is more challenging. Next year, the Mirror DBMS should be ready to participate in the large WEB track.

Acknowledgements

Many thanks go to Peter Boncz and the other members of the CWI Monet team, for their great support. The work reported in this paper is funded in part by the Dutch Telematics Institute project DRUID.

References

- [Bau97] C. Baumgarten. A probabilistic model of distributed information retrieval. In *Proceedings of the 20th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 258–266, 1997.
- [BK95] P.A. Boncz and M.L. Kersten. Monet: An impressionist sketch of an advanced database system. In *BIWIT'95: Basque international workshop on information technology*, July 1995.
- [BMK99] P.A. Boncz, S. Manegold, and M.L. Kersten. Database architecture optimized for the new bottleneck: Memory access. In *Proceedings of 25th International Conference on Very Large Databases (VLDB '99)*, Edinburgh, Scotland, UK, September 1999. To appear.
- [Bro95] E.W. Brown. *Execution performance issues in full-text information retrieval*. PhD thesis, University of Massachusetts, Amherst, October 1995. Also appears as technical report 95-81.

- [CLC95] J.P. Callan, Z. Lu, and W.B. Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 21–28, 1995.
- [dV98] A.P. de Vries. Mirror: Multimedia query processing in extensible databases. In *Proceedings of the fourteenth Twente workshop on language technology (TWLT14): Language Technology in Multimedia Information Retrieval*, pages 37–48, Enschede, The Netherlands, December 1998.
- [dV99] A.P. de Vries. *Content and multimedia database management systems*. PhD thesis, University of Twente, Enschede, The Netherlands, December 1999.
- [dVvDBA99] A.P. de Vries, M.G.L.M. van Doorn, H.M. Blanken, and P.M.G. Apers. The Mirror MMDBMS architecture. In *Proceedings of 25th International Conference on Very Large Databases (VLDB '99)*, Edinburgh, Scotland, UK, September 1999. Technical demo.
- [dVW99] A.P. de Vries and A.N. Wilschut. On the integration of IR and databases. In *Database issues in multimedia; short paper proceedings, international conference on database semantics (DS-8)*, Rotorua, New Zealand, January 1999.
- [Hie99] D. Hiemstra. A probabilistic justification for using tfidf term weighting in information retrieval. *International Journal on Digital Libraries*, 1999. To appear.
- [HK99] D. Hiemstra and W. Kraaij. Twenty-One at TREC-7: Ad-hoc and cross-language track. In Voorhees and Harman [VH99b].
- [HT98] Laura M. Haas and Ashutosh Tiwary, editors. *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*. ACM Press, 1998.
- [Hul93] D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*, pages 329–338, 1993.
- [JFS98] B.Th. Jónsson, M.J. Franklin, and D. Srivastava. Interaction of query evaluation and buffer management for information retrieval. In Haas and Tiwary [HT98], pages 118–129.
- [Ses98] P. Seshadri. Enhanced abstract data types in object-relational databases. *The VLDB Journal*, 7(3):130–140, 1998.
- [STA98] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating mining with relational database systems: Alternatives and implications. In Haas and Tiwary [HT98], pages 343–354.
- [VF95] C.L. Viles and J.C. French. Dissemination of collection wide information in a distributed information retrieval system. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'95)*, pages 167–174, 1995.
- [VGJL95] E.M. Voorhees, N.K. Gupta, and B. Johnson-Laird. The collection fusion problem. In D.K. Harman, editor, *Proceedings of the 3rd Text Retrieval Conference TREC-3*, number 500-225 in NIST Special Publications, pages 95–104, 1995.
- [VH99a] E.M. Voorhees and D.K. Harman. Overview of the 7th text retrieval conference. In *Proceedings of the 7th Text Retrieval Conference TREC-7* [VH99b], pages 1–23.
- [VH99b] E.M. Voorhees and D.K. Harman, editors. *Proceedings of the Seventh Text Retrieval Conference (TREC-7)*, number 500-242 in NIST Special publications, 1999.
- [XC96] J. Xu and W.B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval (SIGIR '96)*, pages 4–11, Zürich, Switzerland, 1996.

Fast Automatic Passage Ranking*

(MultiText Experiments for TREC-8)

G. V. Cormack*

C. L. A. Clarke*

C. R. Palmer⁺

D. I. E. Kisman*

* Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada

⁺ School of Computer Science, Carnegie Mellon University, Pittsburgh

1 Introduction

TREC-8 represents the fifth year that the MultiText project has participated in TREC [2, 1, 4, 5].

The MultiText project develops and prototypes scalable technologies for parallel information retrieval systems implemented on networks of workstations. Research issues are addressed in the context of this parallel architecture. Issues of concern to the MultiText Project include data distribution, load balancing, fast update, fault tolerance, document structure, relevance ranking, and user interaction.

The MultiText system incorporates a unique technique for arbitrary passage retrieval. Since our initial participation in TREC-4 our TREC work has explored variants of this technique.

For TREC-8 we focused our efforts on the Web track. In addition, we submitted runs for the Adhoc task (title and title+description) and a run for the Question Answering task.

2 Arbitrary Passage Retrieval

All our experiments are based on similar passage retrieval techniques. Passages identified by this technique are used in different ways in different experiments. Versions of the technique have been described elsewhere [4, 3]. This section provides a brief, up-to-date description of the technique as used in our TREC-8 experiments.

Each document D in a database is treated as an ordered sequence of terms

$$D = d_1 d_2 d_3 \dots d_m,$$

*Email regarding this article may be sent to mt@plg.uwaterloo.ca. The MultiText project is funded by Communications and Information Technology Ontario.

and a query is treated a set of terms

$$Q = \{q_1, q_2, q_3, \dots\}.$$

An *extent* (u, v) , with $1 \leq u \leq v \leq m$ is used to represent the subsequence of D beginning at position u and ending at position v

$$d_u \ d_{u+1} \ d_{u+2} \dots d_v.$$

An extent (u, v) *satisfies* a term set $T \subseteq Q$ if the subsequence of D defined by the extent contains all the terms from T . That is,

$$|\{t \mid t \in T \text{ and } t \in \{d_u \ d_{u+1} \dots d_v\}\}| = |T|.$$

An extent (u, v) is a *cover* for T if (u, v) satisfies T and the subsequence corresponding to (u, v) contains no subsequence that also satisfies T . That is, there does not exist an extent (u', v') with either $u < u' \leq v' \leq v$ or $u \leq u' \leq v' < v$ that satisfies T .

We generalize the notion of a cover as follows: An extent (u, v) *n-satisfies* a term set $T \subseteq Q$ if the subsequence of D defined by the extent contains exactly n terms from T . That is,

$$|\{t \mid t \in T \text{ and } t \in \{d_u \ d_{u+1} \dots d_v\}\}| = n.$$

An extent (u, v) is an *n-cover* for T if (u, v) *n-satisfies* T and the subsequence corresponding to (u, v) contains no subsequence that also *n-satisfies* T .

The MultiText System uses a fast algorithm to compute *n-covers* over all documents in a collection [3]. Passages are assigned scores based on their lengths and on the weights assigned to the query terms contained within them. A term t is assigned an IDF-like weight

$$w_t = \log(N/f_t),$$

where f_t is the number of times that t appears in the database and N is the total number of term positions in the database (or equivalently, N is the sum of the lengths of all the documents in the database). A standard IDF weight is not used since in-document frequencies are not stored in the MultiText index.

The weight assigned to a set of terms $T \subseteq Q$ is the sum of the weights assigned to each term in T

$$W(T) = \sum_{t \in T} w_t.$$

A score is assigned to an extent based on its length

$$I(p, q) = \begin{cases} \frac{\mathcal{K}}{q-p+1} & \text{if } q - p + 1 \geq \mathcal{K} \\ 1 & \text{if } q - p + 1 \leq \mathcal{K} \end{cases} \quad (1)$$

where \mathcal{K} is a *cutoff* parameter set to values between 1 and 16. If an extent (u, v) is a *n-cover* for the term set T then it can be assigned a score combining the length of the extent and the weight of the terms contained within it

$$C(T, u, v) = W(T') + |T'| \log(I(u, v)) \quad (2)$$

where $T' \subseteq T$ is the set of terms from T contained in term sequence associated with (u, v) .

Method	S-stem?	Fast Adhoc	Small Web	Large Web
1	yes	0.2143 (0.3410) ¹	0.3066 (0.3620) ²	
2	yes	0.2233 (0.3550)	0.3203 (0.3800)	0.4783 (0.5650) ³
2	no	0.2126 (0.3390)	0.3029 (0.3640)	0.4869 (0.5720) ⁴
2*	no	0.2233 (0.3550)	0.3203 (0.3800)	0.4704 (0.5580) ⁵
*max. 3 terms		¹ uwmt8a0	² uwmt8w0	³ uwmt8lw0 ⁴ uwmt8lw1 ⁵ uwmt8lw2

Figure 1: Average Precision (Precision @20) for Web Track and Fast Adhoc Runs.

3 Web Track and Fast Adhoc

The main objective of our Web Track participation was to achieve very fast high-precision retrieval, particularly on the 100 GB Large Web Track Corpus. The general approach is to derive automatically a tiered query - a sequence of successively weaker sets of search terms, which are applied in order until a sufficient number of distinct documents are retrieved to satisfy the task requirements (1000 documents for Adhoc and Small Web; 20 documents for Large Web).

For efficient retrieval we chose to use very small sets of search terms - three or fewer in most cases, and never more than five. For the Adhoc and Small Web tasks we used the title words; for the Large Web task we used only words from the query with stopwords eliminated. Efficiency (and, serendipitously, precision) is improved when passages are restricted to a maximum length (128 words) and the set of search terms rarely coincide within a passage.

Thus we avoid typical sources of inefficiency for retrieval methods based on a large number of weighted terms: we use only a small set of terms in the first place, and we consider only those passages in which exactly these terms co-occur. In contrast, traditional implementations based on weighted terms involve computing the score of every document or passage that contains any of the terms.

In TREC 7, we demonstrated that very good precision and efficiency can be achieved with manually selected sets of terms. Poorer precision resulted when the sets of terms were selected automatically. For TREC 8, our objective was to improve the precision of automatic term selection in tiered queries.

The official runs for the Adhoc and Small Web tasks were due at NIST several weeks before those for the Large Web task. During this interval, we refined our approach to automatic tiering. For this reason, our official fast Adhoc and Small Web runs use the same method (Method 1) while the Large Web run uses a different method (Method 2).

We later applied Method 2 to the Adhoc and Small Web tasks, using NIST's qrels which were published after the official runs were judged. Although these runs are at a disadvantage relative to official runs, Method 2 achieved better precision in all tasks. Average precision and precision @20 for these runs are summarized in Figure 1.

3.1 Tiering Methods

Both tiering methods approximate the ranking formula $C(T, u, v)$ given above as formula 2. A direct computation of $C(T, u, v)$ was rejected for efficiency reasons: such a computation would require that all passages containing any subset of the terms be evaluated. Instead we fix the set of terms in each tier, and compute $I(p, q)$ for each interval of length 128 or less which contains exactly these terms.

Each tier uses as a set of terms T' , a subset of T , the initial terms. These subsets are ordered so that those likely to contribute to a high value of the ranking formula $C(T, u, v)$ are considered first. Method 1 and Method 2 differ in the ordering of tiers.

We note that formula 2 approximates the (logarithm of the) probability that the terms of T' coincide within a passage of length $v - u + 1$, assuming that the term occurrences are uniformly distributed in the corpus.

Method 1 uses instead the actual number of occurrences of passages containing the terms in T' . That is, Method 1 is a two-pass method: the query is evaluated for all $T' \subseteq T$. For each T' , we compute $N(T')$, the number of passages of length 128 or less that contain all the terms of T' . Tiers are weighted by the formula $\log_2(N/N(T'))$. Tiers are ordered by weight, with tiers having similar weights combined by disjunction. More specifically, the first tier is the one with the highest weight (it will necessarily have $T' = T$) and it is combined with all others whose weight is not more than 1 less. The next tier is chosen and combined with subsequent tiers whose weight is within 1, and so on. Within tiers, documents are ranked by formula 1 with $\mathcal{K} = 4$. Because it requires two passes and examines all passages containing all combinations of the query terms, Method 1 is not suitable for very fast retrieval from large corpora. For this reason, we developed Method 2 for the Large Web task.

Method 2 approximates a priori the number of passages containing the terms of T' , assuming the terms are uniformly distributed. This approximation is effected by substituting in formula 2 representative values of u and v such that $u - v + 1 = 128$; that is, the longest acceptable passage containing the terms of T' . As with Method 1, we use $\mathcal{K} = 4$ in formula 1, and combine tiers whose weights differ by less than 1.

For the Large Web task, we used three variants of Method 2: `uwmt81w0` takes as T the set of query words, unstemmed with stopwords removed; `uwmt81w1` takes as T only the three lowest-frequency 3 words from the query; `uwmt81w2` uses the set of query words, with S-stemming applied to those having plural suffixes. For example the query term "goats" would be expanded to {"goat", "goats"} but the term "cow" would be left unexpanded. This method of S-stemming was also applied to the Adhoc and Small Web runs.

Method 2, with and without S-stemming, was applied to the Adhoc and Small Web tasks, yielding the results in Figure 1. The methods used in each row are identical. It was not feasible to apply Method 1 to the Large Web task - no complete set of judgements is available with which to evaluate precision. The methods of `uwmt81w0` and `uwmt81w2` yield identical results when applied to the other two tasks, as there are at most three terms in the title field, from which we drew T .

We note that Method 2 outperforms Method 1 in all runs. S-stemming, on the other hand, improves performance on Adhoc and Small Web, while degrading performance on Large Web.

3.2 Large Web Performance

The Large Web runs were accomplished on two P2-350 computers; the exact configuration was demonstrated at the SIGIR 99 and TREC 8 conferences. Each system runs four copies of a single-threaded search engine, thus achieving CPU/IO overlap and overlap in access among the four disk drives. Each tier is sent to all engines, and all engines respond with the requested number of documents. These are merged by score to produce the combined result. Our choice of two machines was arbitrary: we wanted to build a portable version that we could demonstrate and the index and data fit easily on two machines. Had we used more computers we could have achieved much faster processing times; we have observed in previous experiments that distributing the data yields a linear improvement in query speed. Eventually, constant communication and query setup costs dominate retrieval time; we estimate these costs to be at least an order of magnitude less than the times reported here.

Execution times for the three Large Web runs are as follows: `uwmt81w0`: 0.841 sec/query; `uwmt81w1`: 0.735 sec/query; `uwmt81w2`: 1.010 sec/query. These times were bettered only by AT&T (0.516 seconds/query; 0.354 P@20) and Fujitsu (0.54 sec/query; 0.507 P@20). The only conclusion that can be drawn is that these times are of the same order, although they are achieved on very different hardware platforms.

`uwmt81w0` and `uwmt81w2` achieved the best average precision and precision @20 of all runs. `uwmt81w1` was edged out by a run from Microsoft/City University (1.62 sec/query; 0.561 P@20).

Hawking et al[6] give full comparative results for the Web Tracks.

4 Other Experiments

4.1 Other Adhoc Experiments

One of our Adhoc runs (`uwmt8a0`) was a repeat of our Small Web Track run using the same ranking method. This run was discussed in Section 3.

For our two other Adhoc runs, our passage retrieval technique was used for query expansion via local feedback. For ranking we used a variant of the Okapi formula [7] — specifically BM25 with $b = 0.6$, $k_1 = 1.5$, $k_2 = 0$ and $k_3 = \infty$. One of these runs (`uwmt8a1`) was based on the topic titles only; the other run (`uwmt8a2`) was based on the title and description.

For each query Q we generated all n -covers for the query for all values of n between 1 and the size of the query $|Q|$. Each n -cover was scored using equation 2 and the passages associated with the best 100 n -covers were used for local feedback.

The local feedback processes discarded those n -covers whose length exceeded a threshold value of 256 words and expanded passages whose length was less than 32 words symmetrically to 32 words. In addition, a term was only considered for local feedback if it appeared in at least two different passages. Since passages may overlap, and the same term position may be part of two different passages, terms were also required to appear in at least two distinct positions. Stopwords were also eliminated from consideration.

	before expansion	after expansion	
title-only	average precision	0.2401	0.2673 (+11.3%) ¹
	precision@5	0.5040	0.5560 (+10.3%) ¹
title+desc	average precision	0.2637	0.2671 (+1.2%) ²
	precision@5	0.5520	0.5280 (-4.3%) ²

¹uwmt8a1

²uwmt8a2

Figure 2: Effects of query expansion.

A score was computed for each term as the sum of the scores of the passages in which it was contained. The top 30 terms were used in the final query.

Figure 2 shows the results of query expansion. We were disappointed in the effects of query expansion. Preliminary experiments with the TREC-7 queries led us to expect improvements of 20% or better. Five other groups achieved a better average precision on their official title-only runs. However, uwmt8a1 achieved the best precision@5 of all title-only runs.

4.2 Question Answering

We submitted a single run (uwmt8qa1) to the Question Answering track. Each question was treated as a query and run using the passage retrieval technique described in Section 4.1. The top five passages were filtered to remove tags and to reduce whitespace, truncated to 250 bytes, and submitted as the “answers”.

The run achieved a mean reciprocal rank of 0.471, the sixth highest of the 41 runs.

References

- [1] Charles L. A. Clarke and Gordon V. Cormack. Interactive substring retrieval. In *Fifth Text REtrieval Conference (TREC-4)*, pages 295–304, Gaithersburg, Maryland, November 1996.
- [2] Charles L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. Shortest substring ranking. In *Fourth Text REtrieval Conference (TREC-4)*, pages 295–304, Gaithersburg, Maryland, November 1995.
- [3] Charles L. A. Clarke, Gordon V. Cormack, and Elizabeth A. Tudhope. Relevance ranking for one to three term queries. In *Fifth RIAO Conference*, pages 388–400, Montreal, June 1997. A version of this paper will appear in *Information Processing and Management*, 2000.
- [4] Gordon V. Cormack, Charles L. A. Clarke, Christopher R. Palmer, and Samuel S. L. To. Passage-based refinement. In *Sixth Text REtrieval Conference (TREC-6)*, pages

303–319, Gaithersburg, Maryland, November 1997. A version of this paper will appear in *Information Processing and Management*, 2000.

- [5] Gordon V. Cormack, Christopher R. Palmer, Michael Van Biesbrouck, and Charles L. A. Clarke. Deriving very short queries for high precision and recall. In *Seventh Text REtrieval Conference (TREC-7)*, pages 121–132, Gaithersburg, Maryland, November 1998.
- [6] David Hawking, Ellen Voorhees, Nick Craswell, and Peter Bailey. Overview of the TREC-8 Web Track. In *Eighth Text REtrieval Conference (TREC-7)*, Gaithersburg, Maryland, November 1999.
- [7] S. E. Robertson, S. Walker, and M. Beaulieu. Okapi at TREC-7. In *Seventh Text REtrieval Conference (TREC-7)*, pages 253–264, Gaithersburg, Maryland, November 1998.

Xerox TREC-8 Question Answering Track Report

David A. Hull

Xerox Research Centre Europe
6 chemin de Maupertuis, 38240 Meylan France
hull@xrce.xerox.com

Abstract

This report describes the Xerox work on the TREC-8 Question Answering Track. We linked together a few basic NLP components (a question parser, a sentence boundary identifier, and a proper noun tagger) with a sentence scoring function and an answer presentation function built specifically for the TREC Q&A task. Our system found the correct 50-byte answer (in the top 5 responses) to 45% of the questions, a quite respectable performance, but with considerable room for improvement. Based on the failure analysis presented in this paper, we can conclude that the system would benefit from having access to a broad range of other NLP technologies, including robust parsing and coreference analysis, or some good heuristic approximations thereof. The system also has a clear need for some semantic resources to help with certain difficult problems, such as finding answers that match the semantic class *X* in *What X?* questions.

1 Introduction

Natural language processing (NLP) techniques have not had a large impact on the information retrieval community in recent years. While NLP techniques are extremely useful for stemming and phrase extraction, linguistic techniques can be easily approximated by adhoc methods which run faster and work about as well, at least for the English language. However, there is growing evidence that this will change as the IR community moves towards methods which analyze document content in more depth.¹ The Question Answering task is an important step in that direction. Question Answering is an interesting challenge for NLP researchers because it is new application for many traditional NLP techniques, such as parsing, coreference analysis, and proper name recognition.

Xerox has a long history of research in information retrieval and natural language processing. The TREC Q&A track gives us a nice opportunity to apply the techniques in our NLP toolbox to a new problem. Unfortunately, the author was only able to devote two weeks of time to the TREC-8 task, so expectations were necessarily modest. Our primary goal this year was to test our proper name tagger, called ThingFinder, for this task. Given these constraints, we were quite happy with our TREC-8 performance. The next section introduces the Xerox TREC-8 question answering system. This is followed by an analysis of the TREC-8 results and a commentary on the future prospects of the system.

¹See: <http://www.infotoday.com/searcher/jan00/feldman.htm> for a nice survey of future prospect in information retrieval.

2 Xerox Question Answering System

The Xerox TREC-8 question answering system consists of the following basic components:

- (1) A question parser
- (2) A sentence boundary identifier
- (3) A sentence scoring function
- (4) A proper noun tagger
- (5) An answer presentation function

The Xerox Q&A system operates on a set of documents retrieved by an independent information retrieval system. For convenience, we worked with the top-ranked document set generated by AT&T's TREC-7 adhoc system and provided as a service by Amit Singhal to Question Answering Track participants. Initially, the question is parsed and typed according to the semantic category of the anticipated answer. The documents are divided into sentences, and each sentence is scored by computing its similarity to the query. The top scoring sentences are then passed to a proper noun tagger, and the tagged elements are compared to the question type. All sentences which do not contain an element that matches the question type are removed. The answer presentation function processes the remaining sentences and generates the final result to be presented to the user.

who	<Person>	where	<Place>	what	<What>
whose	<Person>	when	<Time>	which	<What>
whom	<Person>	how	<How>	why	<Unknown>

Table 1: Mapping from keyword to answer category

2.1 The Question Parser

The question parser attempts to identify the question *type* and any secondary arguments which may be associated with that *type*. We define the question *type* as the general semantic class of the expected answer and attempt to identify the following types:

<Person>, <Place>, <Time>, <Money>, <Number>,
<Quantity>, <Name>, <How>, <What>, <Unknown>

The latter four types correspond to an incomplete resolution of the question category. There are two steps to this process. First, we search for the basic keyword which defines the question type, using the mapping shown in Table 1. The <How>, <What>, and <Number> question types may have an associated secondary argument. The secondary arguments are used to further specify the question type. The query is tagged for part of speech and the secondary arguments are extracted using regular expressions defined over sequences of part of speech tags. For example, we identify the sequence: <How> *Adj* (*Adj* = Adjective). If *Adj* is *long* or *short*, the question type is <Quantity>. If *Adj* is *rich* or *poor*, the question type is <Money>. If the secondary argument is not found or not matched, the question type remains <How>. We follow the same pattern for <What> questions. For example, "<What> cost" and "<What> is the cost" both get mapped to <Money>. In hindsight, we realized that who/whom/whose questions should cover both people

and organizations (e.g. Q169: *Whom did the Chicago Bulls beat in the 1993 championship?* is clearly looking for a basketball team.).

If the question does not have one of the basic keywords, it is usually mapped to <Unknown>. However, certain secondary arguments will define the question type without such a keyword. For example, a request such as: *Find the price of a Jaguar XK8*, will be correctly typed as <MONEY> due to the presence of the word *price*. This allows the system to handle some non-standard question formats. The secondary argument list was created by extracting a few thousand questions from a large corpus and by using our general background knowledge of the English language, and consists of 38 elements. The list was constructed in an afternoon and could certainly be enriched. The current system has no knowledge base and no tools for semantic analysis, so much of the extracted secondary information cannot be used. For example, for Q118: *What two researchers . . .*, the system can recognize that it should be looking for researchers, but doesn't know that a researcher is a <Person>. There is clearly a need for some semantic resources in the system.

2.2 Identifying Sentence Boundaries

Sentence boundaries are recognized by a tokenizer written in *awk*. A word token consists of a string of characters delimited by spaces. A word token which ends in a separator character ("?", ")", etc.) defines a sentence boundary. The "." character also defines a sentence boundary unless the word token appears on a list of 206 common abbreviations or satisfies the following *awk* regular expression:

$$/\^[A-Za-z]\.([A-Za-z]\.)+|[A-Z]\.|[A-Z][bcdfghj-np-tvxz]+\.\.)/$$

The tokenizing routine is applied to each of the top ranked documents to divide it into "sentences". For more information on our approach to word and sentence recognition, see Grefenstette [1].

2.3 Sentence Scoring

Each sentence is scored according to the number of words it has in common with the query, using the following weighting function: proper noun = 1.0, number = 0.8, common noun or unknown word = 0.4, other content word = 0.1. The raw sentence score is the weighted sum of the number of unique query words it contains. Each word is counted only once, so a sentence that has all the content words contained in the query receives the maximum possible score. The question keyword (e.g. who, what, etc.) and function words, as identified by the part of speech tagger, are ignored. The sentence score is normalized by dividing by the maximum observed score. Similarly, the score of the document which contained the sentence is normalized by dividing by the score of the top ranked document. The top ranked document data provided by Amit Singhal for use by the track contains the score of each document.

The final sentence score is:

$$S(s, d) = 0.8 * S_n(s) + 0.2 * S_n(d)$$

where $S_n(s)$ is the normalized sentence score and $S_n(d)$ is the normalized document score. Essentially, the document score is used to break ties between sentences containing an equivalent number of query terms (of which there are often very many). Position is used as a second tie-breaking criterion for sentences with the same score from the same document. Sentences are ranked in the same order they appear in the document. The TREC corpus consists almost entirely of newspaper and newswire texts. News articles tend to be written with the most important information first. In addition, there are likely to be more proper nouns and fewer references in the early part of the

text. We realized in hindsight that the sentence is perhaps too small a unit for matching question words and extracting answers. Since we don't currently have any automatic anaphor or ellipsis resolution technology, it would probably be better to analyze longer passages.

2.4 Proper Noun Tagging

ThingFinder is a proper name tagger developed at Xerox by François Trouilleux. It identifies nine different types of proper names, five of which correspond to semantic classes used by our question answering system: person names = <Person>, location names = <Place>, date expressions = <Time>, monetary expressions = <Money>, and other proper names = <Name>. The other four, not currently used by our question answering system, are: percentages, organization names, events, and legal citations (contracts, treaties, etc.). Trouilleux [2] provides a detailed report on the architecture and specifications of ThingFinder.

The remaining question types: <Number> and <Quantity> are extracted based on regular expressions applied to sequences of part of speech tags. The <Number> tag is simply a number, and can have a secondary argument which identifies the item being enumerated. A <Quantity> is a number followed by a one or more words giving the unit of measure. For example, the question *How many people . . . ?*, will be parsed to the pair (<Number>, people). A <Quantity> describes a distance, length, volume, etc, where the exact unit of measurement is not specified. The <Name> type matches any proper name or unknown word (i.e. a word which does not appear in our lexicon).

The question types <How>, <What>, and <Unknown> are used for incomplete question parses. Depending on the secondary arguments, acceptable answers will include noun phrases, verb phrases, and/or unknown words. Sentences are analyzed by ThingFinder in decreasing order of their score. Only sentences with elements which match the question type are retrained. All other sentences are filtered out. This process continues until five matching sentences have been found or the ranked list of sentences has been exhausted.

2.5 Answer Extraction

The Q&A track has two participation categories: under 50 bytes (= characters) and under 250 bytes. Xerox participated in both categories and defined a unique answer extraction routine for each category. The default presentation for the 50 byte category is the proper noun or character string which was tagged to match the question type. If more than one string was found in a given sentence, the strings are concatenated to create a multi-part answer. If the concatenated string is longer than 50 bytes, the answers are split up again, and some may be passed on to the next lower rank position. Sentences are processed from the top until a set of 5 answer strings has been generated. Sometimes this requires less than five sentences if one sentence contains a lot of possible answers.

For example, for (Q3): *What does the Peugeot company manufacture?*, the second ranked sentence was:

Mr Longuet said Peugeot's decision to stop manufacturing bodies for its 504 utility vehicle from the end of November left the fate of the company in Renault's hands.
(FT934-4706)

The question was parsed to (Peugot company, manufacture, <What>). In this case <What> matches any proper noun or noun phrase, so the system produces the following set of possible answers:

Mr Longuet, decision, bodies, 504 utility vehicle, end, November, fate, Renault, hands

The terms *Peugot* and *company* are removed from the list because they are part of the question. Clearly the concatenation of these strings is longer than 50 bytes, so the answer needs to be pruned. In such cases, priority is given to proper nouns and multi-word noun phrases, leading to a final answer of:

Mr Longuet/504 utility vehicle/November/Renault

Given more time, we would have filtering out proper nouns with tags which are less likely to match the question type. In this way, we might have eliminated *Mr Longuet* as a <Person> and *November* as a unit of <Time>.

We eventually plan to link answer selection to a robust parse of the target language sentence, in order to reduce the set of possible answers. We could then search for sentences with the relation SUBJ(*Peugot company*, manufacture) and extract <X> from a relation DOBJ(manufacture, <X>). Parsing might not always find the exact response (likely answer here: *bodies*), but it would narrow search region so that the correct response would fall in the 50-byte window. Alternatively, we could rank answers according to their proximity to the question terms. A 50-byte answer window is sufficiently generous that parsing may not perform better than more approximate methods. However, parsing alone is unlikely to provide sufficient recall to substantially improve the system. The system will also need to be capable of recognizing noun phrase variants and semantically related verbs. For this example, the response *504 utility vehicle* is correct, though perhaps too specific for the question being asked.

We recognize that squeezing multiple answers into the same text string violates the spirit of the Q&A track, since the goal is to find the single right answer to the question. However, we would like to point out that while the system often lacks the world knowledge to make the correct decision, the same task can be easy for the person who asked the question. A human reader would immediately recognize that *504 utility vehicle* is the only plausible correct answer in the above example. To give another example, consider question (Q25): *Who was the lead actress in the movie "Sleepless in Seattle"?*. The first answer returned by our system is:

Tom Hanks/Meg Ryan/John Grisham (FT933-16459)

An English native speaker knows that Meg Ryan is the only female name in the list, and thus the only possible correct response. The assessors confirmed this supposition by marking this answer correct. The alternative solution, coding substantial semantic knowledge into the system, is a very expensive proposition. However, we do hope to gather some semantic information for future versions of the system, using resources such as WordNet or thesauri automatically generated from the corpus.

We made the decision to return multiple responses in a single text string without much knowledge of the assessment process. The assessors were given the following instructions (excerpt²):

- If the answer string contains the answer plus misc. other stuff, judge YES.
- If the answer string contains the answer plus other text that interferes with recognizing the answer, probably judge NO.

This means that the assessors had a lot of flexibility in judging multiple responses. We believe these instructions are an appropriate and reasonable simulation of user reactions. A partial analysis of the results suggests that we did not benefit from our decision to return multiple answers. In fact, it probably hurt our performance. We only returned multiple answers when they were in the

²TREC-8 Question Answering Track Evaluation - presentation by Ellen Voorhess at the TREC-8 Conference

same sentence. As mentioned previously, the 50-byte limit is sufficiently generous that we would probably have been better off simply picking the most appropriate sub-string from the sentence.

For the 250 byte category, the default presentation is the sentence containing an answer which matches the question type. If the sentence is too long, it must be truncated. The simplest way to do this is to present a 250 byte text string centered on the answer. However, it is quite possible that important context could be lost in this fashion. In addition, since many sentences contained more than one possible correct answer, it was often impossible to find a single consecutive 250-character string which contained all the answers. We chose instead to summarize the sentence by successively deleting the least important words. Words were ranked from least to most important as follows:

- (1) All function words
- (2) Adverbs, adjectives, verbs, common nouns (< 5 characters)
- (3) Adverbs, adjectives, verbs, common nouns (\geq 5 characters)
- (4) Multi-word noun phrases and proper nouns
- (5) Distance from "answers"

Fortunately, the system rarely had to procede further than item (2) in this priority list.

For example, for question (Q116): *Which team won the Super Bowl in 1968?*, the system returned:

With Namath as their leader, the AFL's 1968 New York Jets went into Super Bowl III as an 18-point underdog and won, 16-7, against the NFL champion Baltimore Colts, who, 13-1 that season, had romped past such NFL powers as the Chicago Bears, Vikings, 49ers, Giants and Rams. (LA071790-0057)

which becomes:

Namath - - leader - - AFL's - New York Jets - - Super Bowl III - - 18-point underdog
- - - 16-7, against - NFL champion Baltimore Colts - - - - - season - - romped - - NFL
powers - - Chicago Bears - Vikings - 49ers - Giants - Rams

This approach allowed us to keep all the possible correct responses in the answer string in a relatively readable format. However, we did delete two important words (*1968* and *won*) because they had fewer than five characters. Oops! In hindsight, we realized that the sentence summary algorithm should also retain all words which are part of the question. Our assumption throughout is that users of the system will always have the option to view the corresponding full text with the selected answer highlighted. Therefore, the Q&A system should try to maximize the probability that the answer appears in the summary, rather than trying to maximize the chance that the user can verify that the answer is correct simply by looking at the summary.

3 TREC-8 Results and Failure Analysis

Xerox submitted one run in the 50-byte category and one run in the 250-byte category. The results are presented in Table 2. The columns of the table are: average reciprocal rank (1/rank if answer in top 5, 0 otherwise), number/percent of questions with a correct response in the top 5, and the average rank of the Xerox system compared to all other participants (based on average reciprocal rank). In other words, Xerox's 50-byte run averaged between 8th and 9th place out of a total of 20

Category	Avg. 1/Rank	#/% Correct	Avg. Rank
50-byte	0.317	89 / 45%	8.43 / 20
250-byte	0.453	117 / 59%	9.96 / 25

Table 2: Xerox TREC-8 Q&A Track results summary

runs. Our system is managing to answer about half the questions correctly, a result which puts us slightly above average relative to the other TREC-8 Q&A track participants.

Table 3 breaks down the results by question type. *Who* and *what - location* questions are easiest overall, although Xerox also does well on *where* and *what - name* questions. Not surprisingly, *what - location* questions are much easier than *where* questions because the type of location is specified. However, Xerox does about the same on both categories, because the version of ThingFinder used in these experiments has only one tag for location. In other words, it doesn't distinguish between cities and countries. We have surprising success with *what - name* questions relative to the other participants. We attribute this to the fact that no such questions appear in the training set. Due to lack of time, we ignored the training set of questions entirely. While this strategy is not recommended in general, it may have inadvertently helped us write a more general question parser. This may explain our relative success with the *what - money* and *other* questions as well.

Question Type	Num Q's	XRCE			All Systems		
		#C	%C	ARR	#C	%C	ARR
who	48	23	0.48	0.35	18.0	0.38	0.29
where	21	12	0.57	0.40	<u>7.1</u>	<u>0.34</u>	<u>0.23</u>
when	18	7	0.39	0.31	5.6	0.31	0.21
how - number	19	7	0.37	0.37	6.3	0.33	0.26
how - measure	8	2	0.25	<u>0.07</u>	2.0	0.24	0.16
how - money	4	1	0.25	0.25	0.8	0.21	0.18
what - person	8	3	0.38	0.31	2.2	0.28	0.23
what - time	5	2	0.40	0.20	1.4	0.28	0.21
what - location	16	<u>9</u>	<u>0.56</u>	<u>0.47</u>	<u>7.5</u>	<u>0.47</u>	<u>0.34</u>
what - number	2	0	<u>0.00</u>	<u>0.00</u>	0.3	0.15	0.13
what - money	2	2	1.00	0.50	0.6	<u>0.28</u>	<u>0.19</u>
what - name	13	8	0.62	0.41	<u>4.0</u>	<u>0.31</u>	<u>0.23</u>
what (is) X	23	7	0.30	0.18	5.7	0.25	0.18
other	11	4	0.36	0.16	2.2	<u>0.20</u>	0.14
Total	198	87	0.44	0.32	63.8	0.32	0.24

Table 3: Xerox TREC-8 Q&A Track: question breakdown

We performed a failure analysis on the first 100 questions to learn how the system could be improved, focusing on the 50-byte task. This process consisted of reading the question, the top-ranked sentences, the answers returned by our system, and the correct answers. Based on this information, we tried to determine how our system could be improved or what new technologies would be needed to answer the question correctly. The results of the failure analysis will help us prioritize future improvements to the system. Table 4 summarizes the results for the 64 questions for which the system could give a better answer.

The most frequent problem was with our sentence ranking algorithm, with the usual result being that none of the top-ranked sentences contained a correct answer. Our sentence scoring

Problem/solution	freq.	Problem/solution	freq.
sentence ranking	11	tokenization/normalization	5
coreference	10	question word overlap	4
sentence parsing	9	semantic information	3
multiple answers	9	correct answer	2
no clear solution	8	question parsing	1
proper noun tagger	6		

Table 4: Failure analysis of first 100 questions

algorithm assigns proper nouns a high weight and verbs a low weight. While verbs are often less important than nouns in information retrieval, this is much less true in question answering, so the latter decision turned out to be a mistake. A verb match is often a strong indicator of relevance. Other than that, it is often difficult to tell what went wrong. In some cases, it may be that the IR system did not return the necessary documents in its top 200. We did not test this possibility.

Another frequent problem is references. For example, for the question (Q34): *Where is the actress, Marion Davies, buried?*, the following passage contains the answer:

Actress Marion Davies, mistress of William Randolph Hearst, has been dead for almost 30 years, but someone remembers. Earlier this week, a fan left a red rose on her mausoleum in Hollywood Memorial Park.

The system would have had a much better chance of finding the answer if it knew that *her* refers to Marion Davies. Of course, extending the passage length of the answer to 2-3 sentences represents a cheaper solution to this problem. This is an interesting issue to explore in future experiments.

For nine questions, the system could probably find the correct answer with an accurate parse of the answer sentence. For example:

The price collapse has been particularly painful to Grenada for which nutmeg is the main commodity export.

The main commodity export of Grenada is *nutmeg* not *price collapse*, as we unfortunately suggested³. This does not necessarily mean that our robust parser would always find the correct solution. Many TREC-8 participants approximate parsing with a proximity model, and this might well be equally effective. In addition, there were nine questions where the system returned multiple answers, including the correct one, but received no credit from the judges. A good parser would have found the single correct answer in most of these cases. The issue of sentence analysis has an impact on 18 questions in the first 100, making it the single most important issue to address in future work on the system.

There are eight questions which are sufficiently difficult that it is unclear how we could answer them correctly given the current limitations of the technology. There are six questions where our proper name tagger either made an error or does not have a sufficiently detailed tagset to find the correct answer. There are five questions where we had problems with word tokenization or normalization (stemming). For example, we find no overlap between *Winter Olympics* and *Olympic Games*, because *Olympics* is not lemmatized in the former case.

The system failed on four questions because of an implementation error. We made the obvious decision to ignore proper names and common nouns in the answer which also appear in the question. However, this was implemented in such a way that we also ignored answers which partially

³If the system had a thesaurus which identified nutmeg as a commodity, it could find the correct answer without parsing.

overlapped question words. This means that for question (Q51): *Who is the president of Stanford University?*, we filtered out *Stanford University President Donald Kennedy*. It is not necessarily trivial to implement this correctly, because many proper noun variants which overlap the question should be filtered out. For example, if the question had been, *Who is the president of Stanford?*, the phrase, *the Stanford University President*, would not be a correct response. Most likely, one would want to accept responses where the question words serve as modifiers and not as the head noun.

There are three questions where the system needed semantic information to find the correct response. For example, for question (Q85): *Which former Ku Klux Klan member won an elected office in the U.S.?*, it helps a lot to know that a *Ku Klux Klan member* is a person. For two questions, we disagreed with the assessment, and there was one case where we need to improve the question parser. Looking at the results as a whole, it is clear that there is still a lot of work to do, which comes as no surprise.

4 Commentary

From the previous section, we can conclude that our system would benefit from having a lot more NLP technology. This will be our focus in the immediate future. It remains to be seen whether NLP techniques such as parsing and coreference analysis can be approximated by simpler, more efficient heuristics for the Q&A task. In general, we need to establish an optimized answer scoring function which weighs the plausibility of each answer by drawing on many different sources of information. We did not have time to do this for TREC-8, and it is unclear whether the training set of questions we were given was large enough to do this effectively anyway. With the 198 new questions provided in TREC-8, it should definitely be possible to train a good answer ranking function for TREC-9. Many TREC-8 participants have already developed large, complex scoring functions. Our current unweighted scoring model is clearly insufficient, since it does not try to measure the relative correctness of each of many possible answers within the same sentence.

At first glance, it is somewhat astonishing that all TREC-8 Q&A track participants use pretty much the same techniques. This has almost never been the case in the past for a first-year track. This means that there is a strong degree of consensus as to how the TREC Q&A task should be addressed given the current technology. However, we must recognize that the TREC task is a corpus-based information extraction problem, which is only a special subtask of the more general question answering problem. In this context, traditional AI solutions which rely on large knowledge bases are much less appropriate. It is interesting to ask how such systems would perform on the TREC-8 task. The answer is most likely, quite badly. This is not necessarily a problem with their technology. Rather, the knowledge bases of these systems are not constructed to answer questions appropriate to the time period covered by the TREC corpus. As TREC consists primarily of news articles, much of the information has a short lifespan. In addition, most questions were constructed directly from the documents, giving corpus-based extraction techniques a huge advantage. In general, our perspective is that corpus-based and knowledge-based techniques are highly complementary. The former techniques are recall-oriented and the latter techniques are precision-oriented. Future open-domain Q&A systems will need to incorporate both kinds of technology in order to be successful. We can already see this pattern emerging in the evolving market for Web search engine and directory service technology.

References

- [1] G. Grefenstette and P. Tapanainen. What is a word, what is a sentence? Problems of tokenization. In *Proc. of the 3rd Conference of Computational Lexicography and Text Research (COMPLEX '94)*, 1994.
- [2] Francois Trouilleux. Thingfinder prototype english version 2.0. Technical report, Xerox Research Centre Europe - Grenoble, April 1998.

TREC-8 Results

APPENDIX A

This appendix contains the evaluation results for the TREC-8 runs. The initial pages list each of the runs (identified by the run tags) that were included in the different tasks/tracks. Associated with each tag is the organization that produced the run and additional information such as whether the queries were produced manually or automatically as appropriate. Following the run list is a description of the evaluation measures used for the main tasks and many of the tracks. When a track uses different measures, the evaluation measures are described in the track report. The remainder of the appendix contains the evaluation results themselves, in the order given in the run list.

ADHOC RUNS

<u>Tag</u>	<u>Organization</u>	<u>Query Method</u>	<u>Topic Length</u>
acsys8aln2	ACSys	automatic	T+D+N
acsys8alo	ACSys	automatic	T+D+N
acsys8alo2	ACSys	automatic	T+D+N
acsys8amn	ACSys	automatic	T+D
acsys8asn	ACSys	automatic	T
att99atc	AT&T Labs Research	automatic	T
att99atdc	AT&T Labs Research	automatic	T+D
att99atde	AT&T Labs Research	automatic	T+D
att99ate	AT&T Labs Research	automatic	T
weaver1	Carnegie Mellon University	automatic	T+D+N
weaver2	Carnegie Mellon University	automatic	T+D
plt8ah1	City University/Microsoft	automatic	T
plt8ah2	City University/Microsoft	automatic	T+D
plt8ah3	City University/Microsoft	automatic	T+D+N
plt8ah4	City University/Microsoft	automatic	T
plt8ah5	City University/Microsoft	automatic	T+D
Dm8NbnR	Dartmouth College	automatic	T+D
Dm8TFidf	Dartmouth College	automatic	T+D+N
Dm8TFbn	Dartmouth College	automatic	T+D+N
Dm8Nbn	Dartmouth College	automatic	T+D+N
fub99tf	Fondazione Ugo Bordoni	automatic	T+D+N
fub99a	Fondazione Ugo Bordoni	automatic	T+D+N
fub99td	Fondazione Ugo Bordoni	automatic	T+D
fub99tt	Fondazione Ugo Bordoni	automatic	T+D+N
Flab8atdn	Fujitsu Laboratories. Ltd.	automatic	T+D+N
Flab8as	Fujitsu Laboratories. Ltd.	automatic	T+D
Flab8atd2	Fujitsu Laboratories. Ltd.	automatic	T+D
Flab8ax	Fujitsu Laboratories. Ltd.	automatic	T+D+N
Flab8at	Fujitsu Laboratories. Ltd.	automatic	T
GE8ATDN1	GE/Rutgers/SICS/UHelsinki/UPenn	automatic	T+D+N
GE8ATDN2	GE/Rutgers/SICS/UHelsinki/UPenn	automatic	T+D+N
GE8ATD3	GE/Rutgers/SICS/UHelsinki/UPenn	automatic	T+D
ibmg99a	IBM T. J. Watson Research Center (Chong)	automatic	T+D
ibmg99b	IBM T. J. Watson Research Center (Chong)	automatic	T+D
ibmg99c	IBM T. J. Watson Research Center (Chong)	automatic	T+D
ibms99a	IBM T.J. Watson Research Center (Franz)	automatic	T+D
ibms99c	IBM T.J. Watson Research Center (Franz)	automatic	D
ibms99b	IBM T.J. Watson Research Center (Franz)	automatic	T
iit99au1	IIT/AAT/NCR	automatic	T+D
iit99au2	IIT/AAT/NCR	automatic	T+D
ic99dafb	Imperial College	automatic	T+D

ADHOC RUNS (Continued)

<u>Tag</u>	<u>Organization</u>	<u>Query Method</u>	<u>Topic Length</u>
Mer8Adtd1	IRIT/SIG	automatic	T+D
Mer8Adtd2	IRIT/SIG	automatic	T+D
Mer8Adtnd3	IRIT/SIG	automatic	T+D+N
Mer8Adtd4	IRIT/SIG	automatic	T+D
apl8c221	Johns Hopkins University	automatic	T+D+N
apl8n	Johns Hopkins University	automatic	T+D+N
apl8c621	Johns Hopkins University	automatic	T+D+N
apl8p	Johns Hopkins University	automatic	T+D+N
apl8ctd	Johns Hopkins University	automatic	T+D
kuadhoc	Kasetsart University	automatic	T+D+N
kdd8ps16	KDD R&D Laboratories	automatic	T+D+N
kdd8qe01	KDD R&D Laboratories	automatic	T+D+N
kdd8sh16	KDD R&D Laboratories	automatic	T+D
ok8amxc	Microsoft Research Ltd	automatic	T+D
ok8asxc	Microsoft Research Ltd	automatic	T
ok8alx	Microsoft Research Ltd	automatic	T+D+N
MITSLStdn	MIT Laboratory for Computer Science	automatic	T+D+N
MITSLStd	MIT Laboratory for Computer Science	automatic	T+D
uwmt8a0	MuliText Project	automatic	T
uwmt8a1	MuliText Project	automatic	T
uwmt8a2	MuliText Project	automatic	T+D
nttd8ale	NTT DATA Corporation	automatic	T+D+N
nttd8alx	NTT DATA Corporation	automatic	T+D+N
nttd8al	NTT DATA Corporation	automatic	T+D+N
nttd8ame	NTT DATA Corporation	automatic	T+D
nttd8am	NTT DATA Corporation	automatic	T+D
pir9Attd	Queens College, CUNY	automatic	T+D
pir9Aatd	Queens College, CUNY	automatic	T+D+N
pir9Atd0	Queens College, CUNY	automatic	T+D
pir9Aa1	Queens College, CUNY	automatic	T+D+N
pir9At0	Queens College, CUNY	automatic	T
ric8tpn	RICOH Co., Ltd.	automatic	T
ric8dpn	RICOH Co., Ltd.	automatic	T+D
ric8dpx	RICOH Co., Ltd.	automatic	T+D
ric8dnx	RICOH Co., Ltd.	automatic	T+D
ric8tpx	RICOH Co., Ltd.	automatic	T
mds08a3	RMIT	automatic	T+D+N
mds08a2	RMIT	automatic	T+D
mds08a1	RMIT	automatic	T
mds08a4	RMIT	automatic	T+D
mds08a5	RMIT	automatic	T
AntHoc1	Rutgers University	automatic	T+D

ADHOC RUNS (Continued)

<u>Tag</u>	<u>Organization</u>	<u>Query Method</u>	<u>Topic Length</u>
Sab8A1	SabIR Research/Cornell University	automatic	T
Sab8A2	SabIR Research/Cornell University	automatic	D
Sab8A3	SabIR Research/Cornell University	automatic	T+D+N
Sab8A4	SabIR Research/Cornell University	automatic	T+D
Scai8Adhoc	Seoul National University	automatic	T+D
UB99SW	State University of New York at Buffalo	automatic	T+D
UB99T	State University of New York at Buffalo	automatic	T+D
tno8d4	TwentyOne	automatic	T+D
tno8d3	TwentyOne	automatic	T+D
tno8t2	TwentyOne	automatic	T
UniNET8Lg	Universite de Neuchatel	automatic	T+D+N
UniNET8St	Universite de Neuchatel	automatic	T+D
umd99a1	University of Maryland, College Park	automatic	T+D
INQ601	University of Massachusetts	automatic	T
INQ602	University of Massachusetts	automatic	D
INQ603	University of Massachusetts	automatic	T+D
INQ604	University of Massachusetts	automatic	T+D+N
isa25	University of North Carolina (Newby)	automatic	T+D
isa50	University of North Carolina (Newby)	automatic	T+D
isa25t	University of North Carolina (Newby)	automatic	T
isa50t	University of North Carolina (Newby)	automatic	T
unc8a132	University of North Carolina (Yang)	automatic	T+D+N
unc8a142	University of North Carolina (Yang)	automatic	T+D+N
unc8a152	University of North Carolina (Yang)	automatic	T+D
1	University of North Texas	automatic	D
surfahi1	University of Surrey	automatic	T+D+N
surfahi2	University of Surrey	automatic	T+D
surffal2	University of Surrey	automatic	T+D
UT810	University of Twente	automatic	T+D
UT800	University of Twente	automatic	T+D
UT803	University of Twente	automatic	T+D
UT803b	University of Twente	automatic	T+D
UT813	University of Twente	automatic	T+D

ADHOC RUNS (Continued)

<u>Tag</u>	<u>Organization</u>	<u>Query Method</u>
cirtc82	Center for Information Research, Russia	manual
CL99XT	CLARITECH Corporation	manual
CL99SD	CLARITECH Corporation	manual
CL99SDopt1	CLARITECH Corporation	manual
CL99SDopt2	CLARITECH Corporation	manual
CL99XTopt	CLARITECH Corporation	manual
8manexT3D1N0	GE/Rutgers/SICS/UHelsinki/UPenn	manual
GE8MTD2	GE/Rutgers/SICS/UHelsinki/UPenn	manual
iit99ma1	IIT/AAT/NCR	manual
READWARE2	Management Information Technologies, Inc.	manual
READWARE	Management Information Technologies, Inc.	manual
orcl99man	Oracle	manual
discol	Rutgers University	manual

CROSS-LANGUAGE TRACK

<u>Tag</u>	<u>Organization</u>	<u>Topic Language</u>	<u>Run Type</u>
CLARITdmwf	CLARITECH Corporation	english	automatic, T+D+N
CLARITrmwf1	CLARITECH Corporation	english	automatic, T+D+N
CLARITrmwf2	CLARITECH Corporation	english	automatic, T+D+N
CLARITrmwf3	CLARITECH Corporation	english	automatic, T+D+N
EIT99mta	Eurospider Information Technology	german	automatic, T+D+N
EIT99sal	Eurospider Information Technology	german	automatic, T+D
EIT99sta	Eurospider Information Technology	german	automatic, T+D+N
ibmcl8ea	IBM T.J. Watson (Franz)	english	automatic, T+D+N
ibmcl8ec	IBM T.J. Watson (Franz)	english	automatic, T+D+N
ibmcl8fa	IBM T.J. Watson (Franz)	french	automatic, T+D+N
ibmcl8fc	IBM T.J. Watson (Franz)	french	automatic, T+D+N
Mer8Can2x	IRIT/SIG	english	automatic, T+D+N
Mer8Can2x0	IRIT/SIG	english	automatic, T+D+N
Mer8Cfr2x	IRIT/SIG	french	automatic, T+D+N
apxl1	Johns Hopkins University	english	automatic, T+D+N
apxl2	Johns Hopkins University	english	automatic, T+D+N
apxl3	Johns Hopkins University	english	automatic, T+D+N
apxl4	Johns Hopkins University	english	automatic, T+D+N
nmsui1	New Mexico State University	english	manual, T+D
tno8dis	TwentyOne	english	automatic, T+D+N
tno8dpx	TwentyOne	english	automatic, T+D+N
tno8gr	TwentyOne	english	automatic, T+D+N
umd99b1	Univ of Maryland, College Park	english	automatic, T+D+N
umd99b2	Univ of Maryland, College Park	english	automatic, T+D+N
umd99b3	Univ of Maryland, College Park	english	automatic, T+D+N
umd99c1	Univ of Maryland, College Park	english	automatic, T
umd99c2	Univ of Maryland, College Park	english	automatic, T

CROSS-LANGUAGE TRACK – Alternate Tasks

<u>Tag</u>	<u>Organization</u>	<u>Task</u>
sleI2Ef1	Sharp Laboratories	Italian topics against English docs
sleI2Etd1	Sharp Laboratories	Italian topics against English docs
sleI2Et1	Sharp Laboratories	Italian topics against English docs
TW8F2E	TextWise, Inc.	French topics against English docs
TW8E2F	TextWise, Inc.	English topics against French docs
tno8mx	TwentyOne	Merged monolingual run
RaliWebE2EF	Universite de Montreal	English topics against English, French docs
RaliWebF2EF	Universite de Montreal	French topics against English, French docs
RaliHanE2EF	Universite de Montreal	English topics against English, French docs
RaliHanF2EF	Universite de Montreal	French topics against English, French docs

CROSS-LANGUAGE TRACK – GIRT

<u>Tag</u>	<u>Organization</u>	<u>Topic Language</u>	<u>Run Type</u>
EIT99geg	Eurospider Information Technology	English	automatic
EIT99gfg	Eurospider Information Technology	French	automatic
EIT99gmt	Eurospider Information Technology	French	automatic
BKCLGR01	University of California, Berkeley	English	automatic
BKCLGR02	University of California, Berkeley	English	automatic
BKCLGR03	University of California, Berkeley	English	automatic
BKCLGR04	University of California, Berkeley	English	automatic
BKCLGR05	University of California, Berkeley	English	automatic

FILTERING TRACK

<u>Tag</u>	<u>Organization</u>	<u>Run Type</u>	<u>Optimization Measure</u>
plt8f1	City University/Microsoft	batch	LF1
plt8f2	City University/Microsoft	batch	LF1
plt8r1	City University/Microsoft	routing	
plt8r2	City University/Microsoft	routing	
CL99bfL1	CLARITECH Corporation	batch	LF1
CL99bfL2	CLARITECH Corporation	batch	LF2
CL99afL1a	CLARITECH Corporation	adaptive	LF1
CL99afL1b	CLARITECH Corporation	adaptive	LF1
CL99afL1c	CLARITECH Corporation	adaptive	LF1
CL99afL1d	CLARITECH Corporation	adaptive	LF1
CL99afL2	CLARITECH Corporation	adaptive	LF2
CL99afN1	CLARITECH Corporation	adaptive	NF1
dso99rt1	DSO National Laboratories, Singapore	routing	
dso99rt2	DSO National Laboratories, Singapore	routing	
S2N2	Informatique-CDC	routing	
kdd8f001	KDD R&D Laboratories	adaptive	LF1
kdd8f002	KDD R&D Laboratories	adaptive	LF1
kdd8f003	KDD R&D Laboratories	adaptive	LF1
ok8f112	Microsoft Research Ltd	adaptive	LF1
ok8f311	Microsoft Research Ltd	adaptive	LF1
ok8f312	Microsoft Research Ltd	adaptive	LF1
ok8f122	Microsoft Research Ltd	adaptive	LF2
ok8f222	Microsoft Research Ltd	adaptive	LF2
ok8f321	Microsoft Research Ltd	adaptive	LF2
pir9LF1	Queens College, CUNY	adaptive	LF1
pir9LF1a	Queens College, CUNY	adaptive	LF1
pir9LF2	Queens College, CUNY	adaptive	LF2
pir9LF2a	Queens College, CUNY	adaptive	LF2
pirc9BF1	Queens College, CUNY	batch	LF1
pirc9BF2	Queens College, CUNY	batch	LF2
pirc9R1	Queens College, CUNY	routing	
pirc9R2	Queens College, CUNY	routing	LF2
AntBatch1	Rutgers University	batch-adaptive	LF1
AntAdaptive1	Rutgers University	adaptive	LF1
Mer8BaLF1	IRIT/SIG	batch	LF1
Mer8BaLF2	IRIT/SIG	batch	LF2
Mer8BaNF1	IRIT/SIG	batch	NF1
Mer8R1	IRIT/SIG	routing	LF1
Mer8R2	IRIT/SIG	routing	LF1
Scai8Ft	Seoul National University	batch	LF1
uttno8lfl	TwentyOne	adaptive	LF1
uttno8lflf	TwentyOne	adaptive	LF1

FILTERING TRACK

<u>Tag</u>	<u>Organization</u>	<u>Run Type</u>	<u>Optimization Measure</u>
uttno8lf1p	TwentyOne	adaptive	LF2
uttno8lf2	TwentyOne	adaptive	LF2
uttno8lf2f	TwentyOne	adaptive	LF2
uttno8lf2p	TwentyOne	adaptive	LF2
IOWAF992	University of Iowa	adaptive	LF1
IOWAF993	University of Iowa	adaptive	LF1
IOWAF991	University of Iowa	adaptive	LF2
INQ610	University of Massachusetts	adaptive	LF1
INQ613	University of Massachusetts	adaptive	LF1
INQ611	University of Massachusetts	adaptive	LF2
INQ612	University of Massachusetts	adaptive	LF2
umrlsi	University of Maryland, College Park	routing	
umrqz	University of Maryland, College Park	routing	

INTERACTIVE TRACK

<u>Tag</u>	<u>Organization</u>
nmsui	New Mexico State University
ohsui	Oregon Health Sciences University
rmiti	RMIT
ruti	Rutgers University
berki	University of California, Berkeley
unci	University of North Carolina (Yang)
shefi	The University of Sheffield, UK

QUESTION ANSWERING TRACK

<u>Tag</u>	<u>Organization</u>	<u>Maximum Answer Length</u>
attqa50e	AT&T Labs Research	50
attqa250e	AT&T Labs Research	250
attqa50p	AT&T Labs Research	50
attqa250p	AT&T Labs Research	250
clr99s	CL Research	250
textract9908	Cymfony Inc.	50
GePenn	GE/Rutgers/SICS/UHelsinki/UPenn	250
CRDBASE050	GE/Rutgers/SICS/UHelsinki/UPenn	50
CRDBASE250	GE/Rutgers/SICS/UHelsinki/UPenn	250
IBMDR995	IBM T. J. Watson Research Center (Chong)	50
IBMDR992	IBM T. J. Watson Research Center (Chong)	250
IBMVS995	IBM T. J. Watson Research Center (Chong)	50
IBMVS992	IBM T. J. Watson Research Center (Chong)	250
LimsiLC	LIMSI-CNRS	250
MTR99050	MITRE	50
MTR99250	MITRE	250
uwmt8qa1	MuliText Project	250
NTU99	National Taiwan University	250
CRL50	New Mexico State University	50
CRL250	New Mexico State University	250
nttd8ql1	NTT DATA Corporation	250
nttd8ql4	NTT DATA Corporation	250
nttd8qs1	NTT DATA Corporation	50
nttd8qs2	NTT DATA Corporation	50
mds08q1	RMIT	250
Scai8QnA	Seoul National University	250
SMUNLP1	Southern Methodist University	50
SMUNLP2	Southern Methodist University	250
UIowaQA1	University of Iowa	250
UIowaQA2	University of Iowa	250
UIowaQA3	University of Iowa	50
UIowaQA4	University of Iowa	50
umdqa	University of Maryland, College Park	50
INQ634	University of Massachusetts	50
INQ635	University of Massachusetts	250
INQ638	University of Massachusetts	50
INQ639	University of Massachusetts	250
UOandNRC	University of Ottawa	250
UOandNRC50	University of Ottawa	50
shefatt50	The University of Sheffield, UK	50
shefatt250	The University of Sheffield, UK	250
shefinq50	The University of Sheffield, UK	50
shefinq250	The University of Sheffield, UK	250
xeroxQA8lC	Xerox Research Centre Europe	250
xeroxQA8sC	Xerox Research Centre Europe	50

SPOKEN DATA RECOGNITION TRACK

<u>Tag</u>	<u>Organization</u>	<u>Language Model</u>	<u>Boundary Condition</u>
att-b1	AT&T Labs Research	rolling	known
att-cr-cmus1	AT&T Labs Research	rolling	known
att-cr-cuhtks1	AT&T Labs Research	rolling	known
att-cr-cuhtks1p1	AT&T Labs Research	rolling	known
att-cr-limsi1	AT&T Labs Research	rolling	known
att-cr-shefs1	AT&T Labs Research	rolling	known
att-r1	AT&T Labs Research	rolling	known
att-s1	AT&T Labs Research	rolling	known
att-s2	AT&T Labs Research	rolling	known
cuhtk-b1	Cambridge University	fixed	known
cuhtk-b1u	Cambridge University	fixed	unknown
cuhtk-cr-att-s1	Cambridge University	fixed	known
cuhtk-cr-b2	Cambridge University	fixed	known
cuhtk-cr-cmu-s1	Cambridge University	fixed	known
cuhtk-cr-limsi-s1	Cambridge University	fixed	known
cuhtk-cr-shef-s1	Cambridge University	fixed	known
cuhtk-cru-limsi-s1	Cambridge University	fixed	unknown
cuhtk-cru-nist-b2	Cambridge University	rolling	unknown
cuhtk-cru-shef-s1	Cambridge University	fixed	unknown
cuhtk-r1	Cambridge University	fixed	known
cuhtk-s1	Cambridge University	fixed	known
cuhtk-s1p1	Cambridge University	fixed	known
cuhtk-s1plu	Cambridge University	fixed	unknown
cuhtk-s1u	Cambridge University	fixed	unknown
cmu-b1	Carnegie Mellon University	rolling	known
cmu-r1	Carnegie Mellon University	rolling	known
cmu-s1	Carnegie Mellon University	rolling	known
cmu-s2	Carnegie Mellon University	rolling	known
ibms-b1	IBM T.J. Watson (Franz)	fixed	known
ibms-r1	IBM T.J. Watson (Franz)	fixed	known
limsi-b1	LIMSI-CNRS	fixed	known
limsi-b2	LIMSI-CNRS	rolling	known
limsi-cr-att1	LIMSI-CNRS	fixed	known
limsi-cr-cmu1	LIMSI-CNRS	fixed	known
limsi-cr-cuhtk1	LIMSI-CNRS	fixed	known
limsi-cr-shef1	LIMSI-CNRS	fixed	known
limsi-r1	LIMSI-CNRS	fixed	known
limsi-s1	LIMSI-CNRS	fixed	known
mds08-b1	RMIT	fixed	known
mds08-r1	RMIT	fixed	known
cedar-b1	State Univ of NY at Buffalo	fixed	known
cedar-r1	State Univ of NY at Buffalo	fixed	known

SPOKEN DATA RECOGNITION TRACK

<u>Tag</u>	<u>Organization</u>	<u>Language Model</u>	<u>Boundary Condition</u>
tno8b-b1-limsi	TwentyOne	fixed	known
tno8b-b1u-limsi	TwentyOne	fixed	unknown
tno8b-r1-limsi	TwentyOne	fixed	known
tno8b-s1-limsi	TwentyOne	fixed	known
tno8b-s1u-limsi	TwentyOne	fixed	unknown
umass-b1	University of Massachusetts	fixed	known
umass-r1	University of Massachusetts	fixed	known
shef-b1	Sheffield/Cambridge/SoftSound/ICSI	fixed	known
shef-b1u	Sheffield/Cambridge/SoftSound/ICSI	fixed	unknown
shef-cr-att-s1	Sheffield/Cambridge/SoftSound/ICSI	fixed	known
shef-cr-cmu-s1	Sheffield/Cambridge/SoftSound/ICSI	fixed	known
shef-cr-cuhtk-s1	Sheffield/Cambridge/SoftSound/ICSI	fixed	known
shef-cr-cuhtk-s1p1	Sheffield/Cambridge/SoftSound/ICSI	fixed	known
shef-cr-limsi-s1	Sheffield/Cambridge/SoftSound/ICSI	fixed	known
shef-cr-nist-b2	Sheffield/Cambridge/SoftSound/ICSI	fixed	known
shef-cru-cuhtk-s1p1u	Sheffield/Cambridge/SoftSound/ICSI	fixed	unknown
shef-cru-cuhtk-s1u	Sheffield/Cambridge/SoftSound/ICSI	fixed	unknown
shef-cru-limsi-s1u	Sheffield/Cambridge/SoftSound/ICSI	fixed	unknown
shef-cru-nist-b2u	Sheffield/Cambridge/SoftSound/ICSI	fixed	unknown
shef-r1	Sheffield/Cambridge/SoftSound/ICSI	fixed	known
shef-s1	Sheffield/Cambridge/SoftSound/ICSI	fixed	known
shef-s1u	Sheffield/Cambridge/SoftSound/ICSI	fixed	unknown

SMALL WEB TRACK

<u>Tag</u>	<u>Organization</u>	<u>Run Type</u>
acsys8wm	ACSys	content-only
acsys8wmp	ACSys	content-link
acsys8wmq	ACSys	content-link
acsys8wmr	ACSys	content-link
att99wtde	AT&T Labs Research	content-only
att99wtde	AT&T Labs Research	content-only
CL99WebH	CLARITECH Corporation	content-only
CL99WebM	CLARITECH Corporation	content-only
DCU99C01	Dublin City University	content-only
DCU99L01	Dublin City University	content-link
DCU99L02	Dublin City University	content-link
Flab8wtdN	Fujitsu Laboratories. Ltd.	content-only
Flab8wtdnN	Fujitsu Laboratories. Ltd.	content-only
iit99wt1	IIT/AAT/NCR	content-only
iit99wt2	IIT/AAT/NCR	content-link
iit99wt3	IIT/AAT/NCR	content-link
Mer8Wci1	IRIT/SIG	content-link
Mer8Wci2	IRIT/SIG	content-link
Mer8Wci3	IRIT/SIG	content-link
Mer8Wctd	IRIT/SIG	content-only
ok8wmx	Microsoft Research Ltd	content-only
uwmt8w0	MuliText Project	content-only
hio1	Oslo College	content-only
hio2	Oslo College	content-link
hio3	Oslo College	content-link
hio4	Oslo College	content-link
mds08w1	RMIT	content-only
mds08w2	RMIT	content-link
mds08w3	RMIT	content-link
disco2	Rutgers University	content-only
disco3	Rutgers University	content-link
Scai8Web1	Seoul National University	content-only
Scai8Web2	Seoul National University	content-link
UniNEWCt	Universite de Neuchatel	content-only
UniNEW2Ct	Universite de Neuchatel	content-only
UniNEWLink	Universite de Neuchatel	content-link
UniNEW2Link	Universite de Neuchatel	content-link
uiowaweb1	University of Iowa	content-only
uiowaweb2	University of Iowa	content-link
INQ620	University of Massachusetts	content-only
isw25	University of North Carolina (Newby)	content-only
isw25t	University of North Carolina (Newby)	content-only
isw50	University of North Carolina (Newby)	content-only
isw50t	University of North Carolina (Newby)	content-only

LARGE WEB TRACK FULL COLLECTION RUNS

<u>Tag</u>	<u>Organization</u>	<u>Run Type</u>	<u>Collection Size</u>
acsys8lw0	ACSys	content-only	100 gB
acsys8lw0_pr1	ACSys	content-link	100 gB
acsys8lw0_pr10	ACSys	content-link	100 gB
att99vlci	AT&T Labs Research	content-only	100 gB
att99vlcm	AT&T Labs Research	content-only	100 gB
fl8wlnsb	Fujitsu Laboratories. Ltd.	content-only	100 gB
fl8wlnsr	Fujitsu Laboratories. Ltd.	content-only	100 gB
fl8wlsb	Fujitsu Laboratories. Ltd.	content-only	100 gB
ok8v1	Microsoft Research Ltd	content-only	100 gB
ok8v2	Microsoft Research Ltd	content-only	100 gB
plt8wt1	Microsoft Research Ltd/CityU	content-only	100 gB
uwmt8lw0	MuliText Project (UWaterloo)	content-only	100 gB
uwmt8lw1	MuliText Project (UWaterloo)	content-only	100 gB
uwmt8lw2	MuliText Project (UWaterloo)	content-only	100 gB
INQ650	University of Massachusetts	content-only	100 gB
iswqd1	University of North Carolina	content-only	100 gB
iswqd2	University of North Carolina	content-only	100 gB

SUBSET RUNS

<u>Tag</u>	<u>Organization</u>	<u>Run Type</u>	<u>Collection Size</u>
plt8wt2	Microsoft Research Ltd/CityU	content-only	10 gB
plt8wt3	Microsoft Research Ltd/CityU	content-only	1 gB
isb1b	University of North Carolina	content-only	1 gB
isb1qd	University of North Carolina	content-only	1 gB

Evaluation Techniques and Measures

Categories

The results following this section are organized according to the task accomplished by the run: ad hoc or a track task. Some tracks do not use these evaluation tools. However, the evaluation tools used are described in the results section for the track.

Ad hoc

Retrieval using an "ad hoc" topic such as a researcher might use in a library environment. In TREC this implies that the input topic has no training material such as relevance judgments to aid in the construction of the input query. Systems ran TREC topics against all documents from TREC Disks 4 (except Congressional Record) and 5.

Evaluation Measures

I. Recall

A measure of the ability of a system to present all relevant items.

$$\text{recall} = \frac{\text{number of relevant items retrieved}}{\text{number of relevant items in collection}}$$

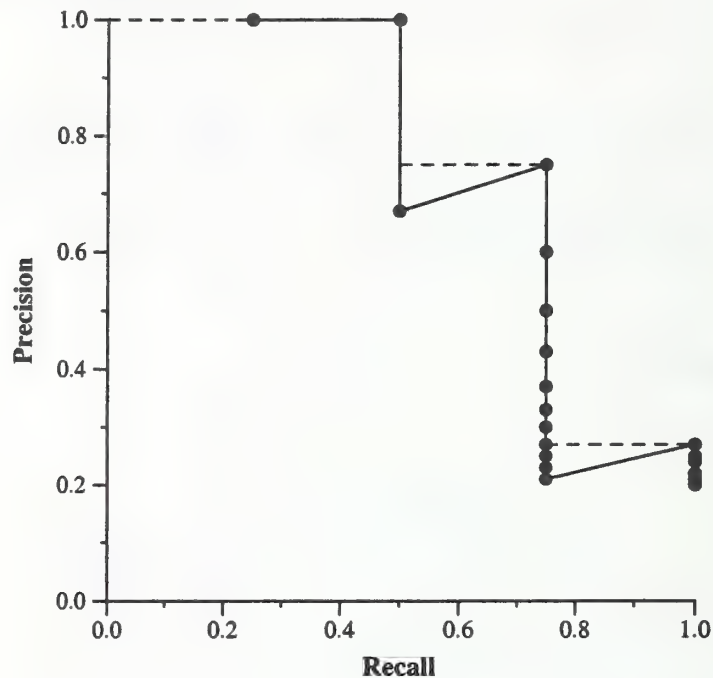
II. Precision.

A measure of the ability of a system to present only relevant items.

$$\text{precision} = \frac{\text{number of relevant items retrieved}}{\text{total number of items retrieved}}$$

Precision and recall are set-based measures. That is, they evaluate the quality of an unordered set of retrieved documents. To evaluate ranked lists, precision can be plotted against recall after each retrieved document as shown in the example below. To facilitate computing average performance over a set of topics, each with a different number of relevant documents, individual topic precision values are interpolated to a set of standard recall levels (0 to 1 in increments of .1). The particular rule used to interpolate precision at standard recall level i is to use the maximum precision obtained for the topic for any actual recall level greater than or equal to i . Note that while precision is not defined at a recall of 0.0, this interpolation rule does define an interpolated value for recall level 0.0. In the example, the actual precision values are plotted with circles (and connected by a solid line) and the interpolated precision is shown with the dashed line.

Example: Assume a document collection has 20 documents, four of which are relevant to topic t . Further assume a retrieval system ranks the relevant documents first, second, fourth, and fifteenth. The exact recall points are 0.25, 0.5, 0.75, and 1.0. Using the interpolation rule, the interpolated precision for all standard recall levels up to .5 is 1, the interpolated precision for recall levels .6 and .7 is .75, and the interpolated precision for recall levels .8 or greater is .27.



System Results Description

Each of the following pages contains the evaluation results for one run. A page is comprised of a header (containing the task and organization name), 3 tables, and 2 graphs.

Tables

Tables are generated by *trec_eval* courtesy of Chris Buckley using the SMART methodology.

I. "Summary Statistics" Table

Table 1 is a sample "Summary Statistics" Table

Table 1: Sample “Summary Statistics” Table.

Summary Statistics	
Run	Cor7A1clt-automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel_ret:	2621

A. Run

A description of the run. It contains the run tag provided by the participant, and as applicable, whether queries were constructed manually or automatically, and whether the title, description, or narrative of the topic was used.

B. Number of Topics

Number of topics searched in this run (generally 50 topics are run for each task).

C. Total number of documents over all topics (the number of topics given in B).

i. Retrieved

Number of documents submitted to NIST. This is usually 50,000 (50 topics \times 1000 documents), but is less when fewer than 1000 documents are retrieved per topic.

ii. Relevant

Total possible relevant documents within a given task and category.

iii. Rel_ret

Total number of relevant documents returned by a run over all the topics.

II. “Recall Level Precision Averages” Table.

Table 2 is a sample “Recall Level Precision Averages” Table.

A. Precision at 11 standard recall levels

The precision averages at 11 standard recall levels are used to compare the performance of different systems and as the input for plotting the recall-precision graph (see below). Each recall-precision average is computed by summing the interpolated precisions at the specified recall cutoff value (denoted by $\sum P_\lambda$ where P_λ is the interpolated precision at recall level λ) and then dividing by the number of topics.

$$\frac{\sum_{i=1}^{NUM} P_\lambda}{NUM} \quad \lambda = \{0.0, 0.1, 0.2, 0.3, \dots, 1.0\}$$

- Interpolating recall-precision

Standard recall levels facilitate averaging and plotting retrieval results.

Table 2: Sample "Recall Level Precision Averages" Table.

Recall Level Precision Averages	
Recall	Precision
0.00	0.6169
0.10	0.4517
0.20	0.3938
0.30	0.3243
0.40	0.2715
0.50	0.2224
0.60	0.1642
0.70	0.1342
0.80	0.0904
0.90	0.0472
1.00	0.0031
Average precision over all relevant docs	
non-interpolated	0.2329

B. Average precision over all relevant documents, non-interpolated

This is a single-valued measure that reflects the performance over all relevant documents. It rewards systems that retrieve relevant documents quickly (highly ranked).

The measure is not an average of the precision at standard recall levels. Rather, it is the average of the precision value obtained after each relevant document is retrieved. (When a relevant document is not retrieved at all, its precision is assumed to be 0.) As an example, consider a query that has four relevant documents which are retrieved at ranks 1, 2, 4, and 7. The actual precision obtained when each relevant document is retrieved is 1, 1, 0.75, and 0.57, respectively, the mean of which is 0.83. Thus, the average precision over all relevant documents for this query is 0.83.

III. "Document Level Averages" Table

Table 3 is a sample "Document Level Averages" Table.

A. Precision at 9 document cutoff values

The precision computed after a given number of documents have been retrieved reflects the actual measured system performance as a user might see it. Each document precision average is computed by summing the precisions at the specified document cutoff value and dividing by the number of topics (50).

B. R-Precision

R-Precision is the precision after R documents have been retrieved, where R is the number of relevant documents for the topic. It de-emphasizes the exact ranking of the retrieved relevant documents, which can be particularly useful in TREC where there are large numbers of relevant documents.

The average R-Precision for a run is computed by taking the mean of the R-Precisions of the individual topics in the run. For example, assume a run consists of two topics,

Table 3: Sample "Document Level Averages" Table.

Document Level Averages	
	Precision
At 5 docs	0.4280
At 10 docs	0.3960
At 15 docs	0.3493
At 20 docs	0.3370
At 30 docs	0.3100
At 100 docs	0.2106
At 200 docs	0.1544
At 500 docs	0.0875
At 1000 docs	0.0524
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2564

one with 50 relevant documents and another with 10 relevant documents. If the retrieval system returns 17 relevant documents in the top 50 documents for the first topic, and 7 relevant documents in the top 10 for the second topic, then the run's R-Precision would be $\frac{\frac{17}{50} + \frac{7}{10}}{2}$ or 0.52.

Graphs

I. Recall-Precision Graph

Figure 1 is a sample Recall-Precision Graph.

The Recall-Precision Graph is created using the 11 cutoff values from the Recall Level Precision Averages. Typically these graphs slope downward from left to right, enforcing the notion that as more relevant documents are retrieved (recall increases), the more nonrelevant documents are retrieved (precision decreases).

This graph is the most commonly used method for comparing systems. The plots of different runs can be superimposed on the same graph to determine which run is superior. Curves closest to the upper right-hand corner of the graph (where recall and precision are maximized) indicate the best performance. Comparisons are best made in three different recall ranges: 0 to 0.2, 0.2 to 0.8, and 0.8 to 1. These ranges characterize high precision, middle recall, and high recall performance, respectively.

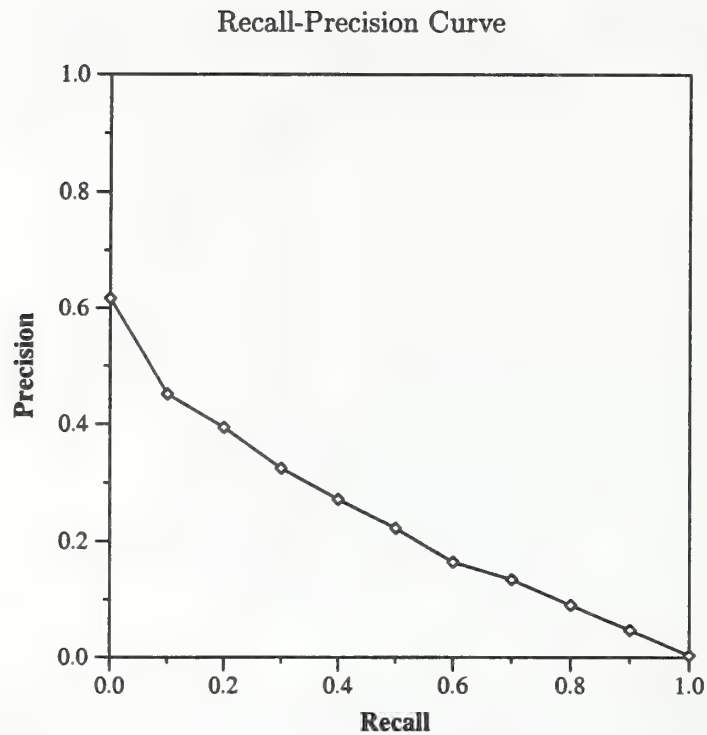


Figure 1: Sample Recall-Precision Graph.

II. Average Precision Histogram.

Figure 2 is a sample Average Precision Histogram.

The Average Precision Histogram measures the average precision of a run on each topic against the median average precision of all corresponding runs on that topic. This graph is intended to give insight into the performance of individual systems and the types of topics that they handle well.

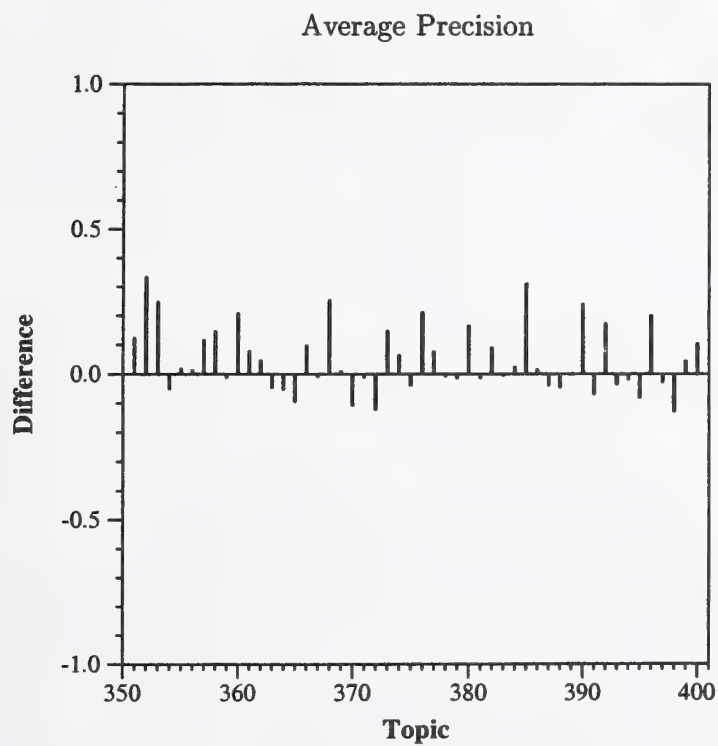
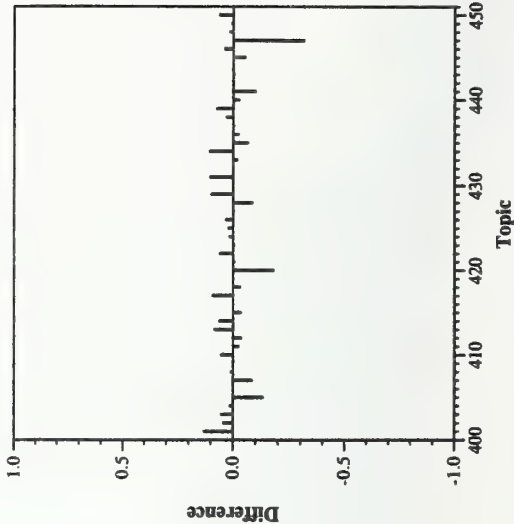
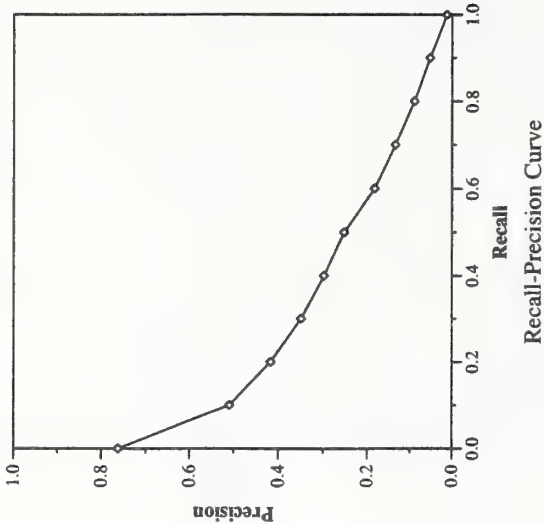


Figure 2: Sample Average Precision Histogram.

Summary Statistics	
Run Number	acsys8aln2
Run Description	Automatic, title + desc + narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3085

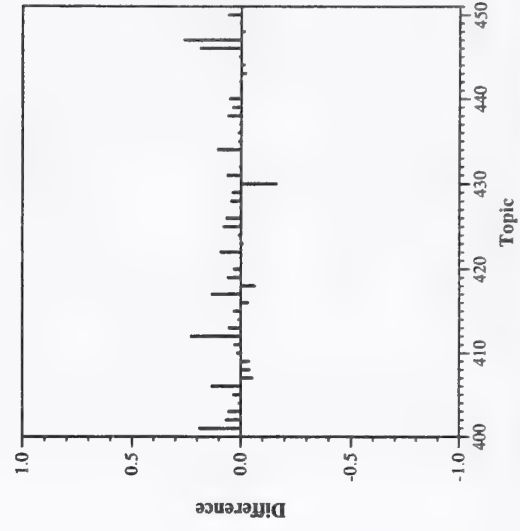
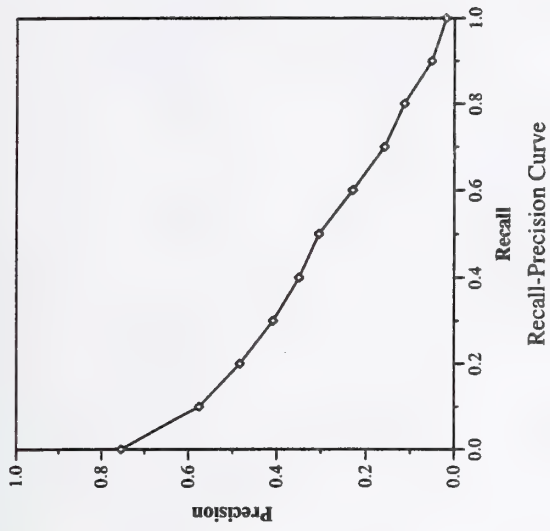
Recall Level Precision Averages	
Recall	Precision
0.00	0.7623
0.10	0.5096
0.20	0.4169
0.30	0.3493
0.40	0.2980
0.50	0.2523
0.60	0.1831
0.70	0.1346
0.80	0.0895
0.90	0.0524
1.00	0.0122
Average precision over all relevant docs	
non-interpolated	0.2560

Document Level Averages	
	Precision
At 5 docs	0.5640
At 10 docs	0.4780
At 15 docs	0.4360
At 20 docs	0.4060
At 30 docs	0.3660
At 100 docs	0.2402
At 200 docs	0.1734
At 500 docs	0.1004
At 1000 docs	0.0617
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3076



Summary Statistics		
Run Number	acsys8alo	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3312	

Recall Level Precision Averages		Document Level Averages	
Recall	Precision		Precision
0.00	0.7544	At 5 docs	0.5560
0.10	0.5754	At 10 docs	0.5300
0.20	0.4832	At 15 docs	0.4720
0.30	0.4078	At 20 docs	0.4400
0.40	0.3495	At 30 docs	0.3993
0.50	0.3050	At 100 docs	0.2586
0.60	0.2298	At 200 docs	0.1876
0.70	0.1588	At 500 docs	0.1102
0.80	0.1141	At 1000 docs	0.0662
0.90	0.0517	R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
1.00	0.0187		
Average precision over all relevant docs		Exact	0.3343



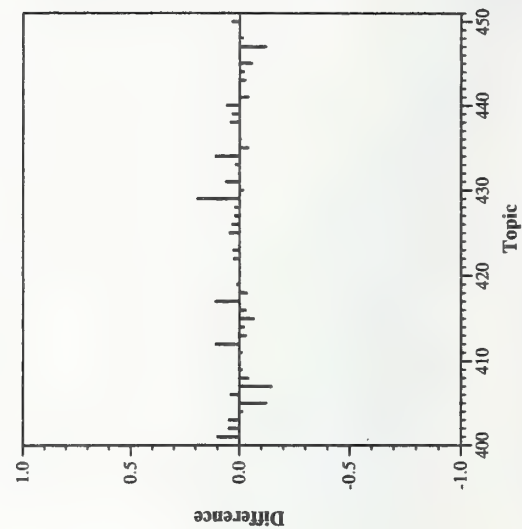
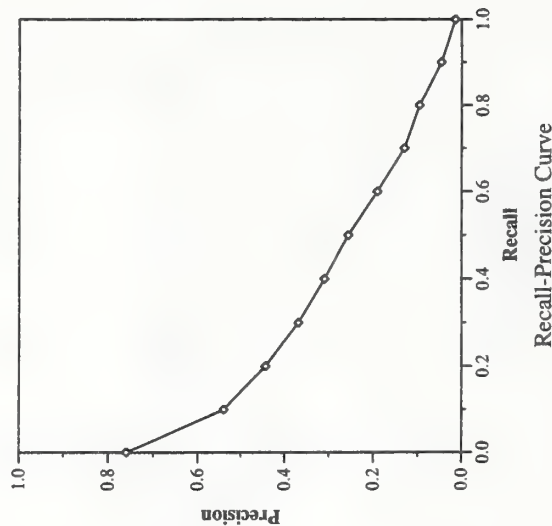
Difference from Median in Average Precision per Topic

Ad hoc results — ACSys

Summary Statistics		
Run Number	acsys8alo2	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3080	

Recall Level Precision Averages		
	Recall	Precision
	0.00	0.7595
	0.10	0.5400
	0.20	0.4438
	0.30	0.3689
	0.40	0.3098
	0.50	0.2563
	0.60	0.1911
	0.70	0.1301
	0.80	0.0960
	0.90	0.0464
	1.00	0.0149
Average precision over all relevant docs		
	non-interpolated	0.2637

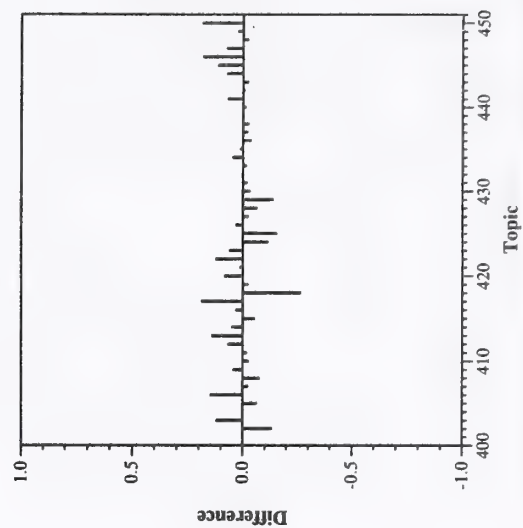
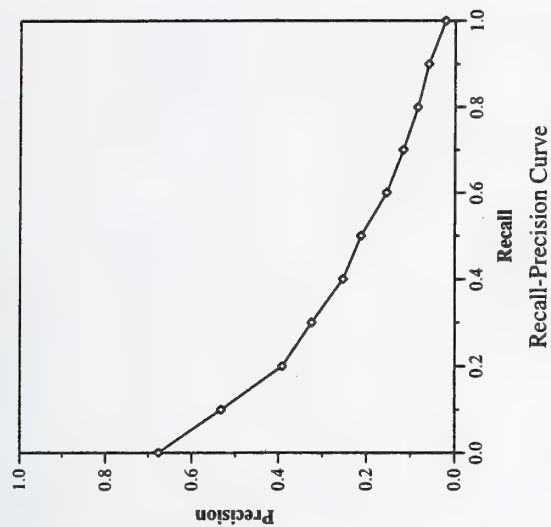
Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.4740
At 15 docs	0.4453
At 20 docs	0.4160
At 30 docs	0.3653
At 100 docs	0.2424
At 200 docs	0.1748
At 500 docs	0.1010
At 1000 docs	0.0616
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3168



Summary Statistics		
Run Number	acsys8amn	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2570	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6764
0.10	0.5331
0.20	0.3915
0.30	0.3251
0.40	0.2535
0.50	0.2122
0.60	0.1543
0.70	0.1165
0.80	0.0841
0.90	0.0597
1.00	0.0216
Average precision over all relevant docs	
non-interpolated	0.2353

Document Level Averages	
	Precision
At 5 docs	0.4600
At 10 docs	0.4380
At 15 docs	0.4093
At 20 docs	0.3790
At 30 docs	0.3427
At 100 docs	0.2144
At 200 docs	0.1466
At 500 docs	0.0820
At 1000 docs	0.0514
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2715

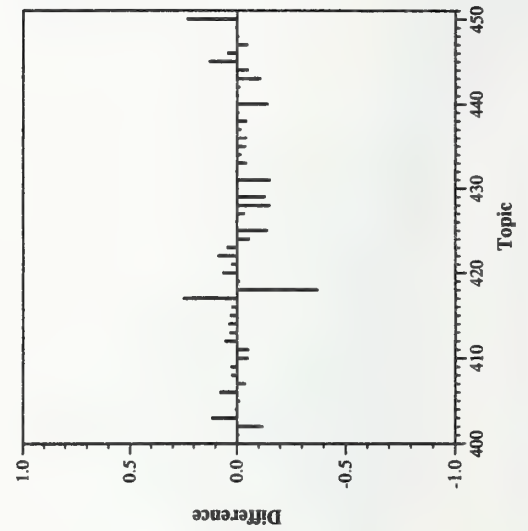
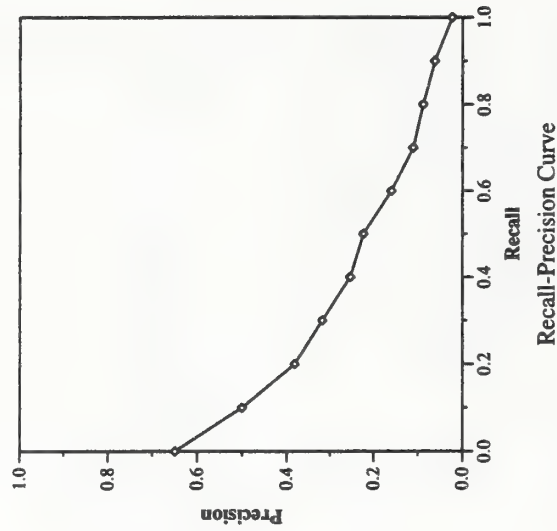


Ad hoc results — ACSys

Summary Statistics	
Run Number	acsys8asn
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	2500

Recall Level Precision Averages	
Recall	Precision
0.00	0.6492
0.10	0.5007
0.20	0.3820
0.30	0.3182
0.40	0.2535
0.50	0.2235
0.60	0.1598
0.70	0.1106
0.80	0.0887
0.90	0.0619
1.00	0.0233
Average precision over all relevant docs	
non-interpolated	0.2309

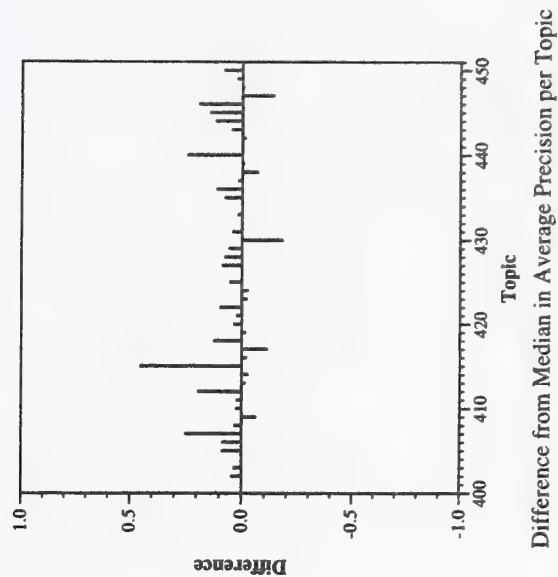
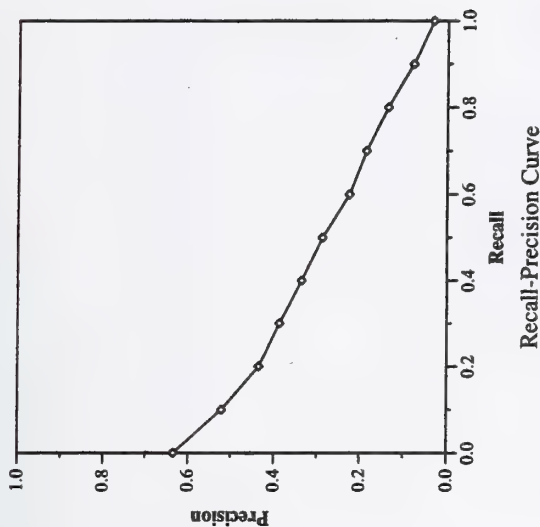
Document Level Averages	
	Precision
At 5 docs	0.4280
At 10 docs	0.4300
At 15 docs	0.4040
At 20 docs	0.3790
At 30 docs	0.3307
At 100 docs	0.2114
At 200 docs	0.1459
At 500 docs	0.0824
At 1000 docs	0.0500
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2687



Summary Statistics		
Run Number	att99atc	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3089	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6336
0.10	0.5214
0.20	0.4350
0.30	0.3871
0.40	0.3362
0.50	0.2878
0.60	0.2255
0.70	0.1856
0.80	0.1363
0.90	0.0785
1.00	0.0324
Average precision over all relevant docs	
non-interpolated	0.2853

Document Level Averages	
	Precision
At 5 docs	0.5160
At 10 docs	0.4740
At 15 docs	0.4267
At 20 docs	0.4160
At 30 docs	0.3760
At 100 docs	0.2514
At 200 docs	0.1819
At 500 docs	0.1045
At 1000 docs	0.0618
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3117

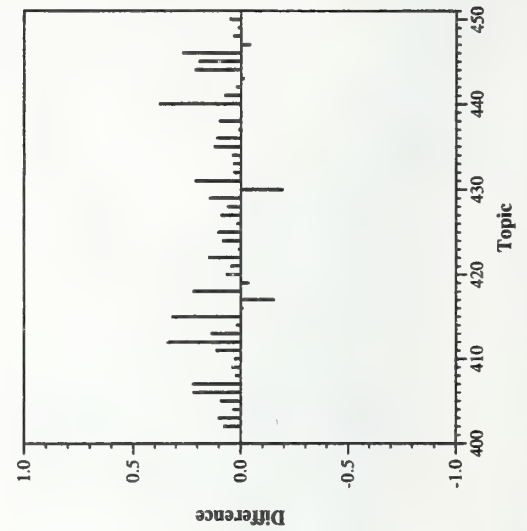
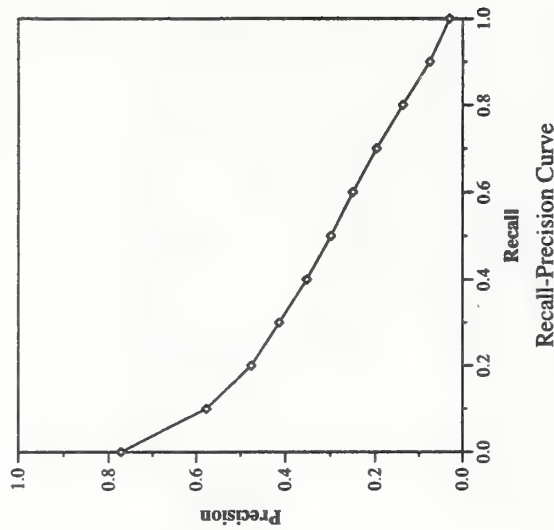


Ad hoc results — AT&T Labs Research

Summary Statistics		
Run Number	att99atdc	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3170	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7721
0.10	0.5772
0.20	0.4760
0.30	0.4137
0.40	0.3527
0.50	0.2991
0.60	0.2507
0.70	0.1975
0.80	0.1378
0.90	0.0755
1.00	0.0306
Average precision over all relevant docs	
non-interpolated	0.3089

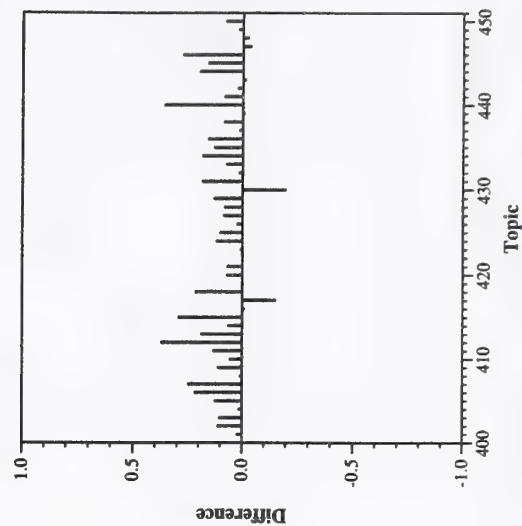
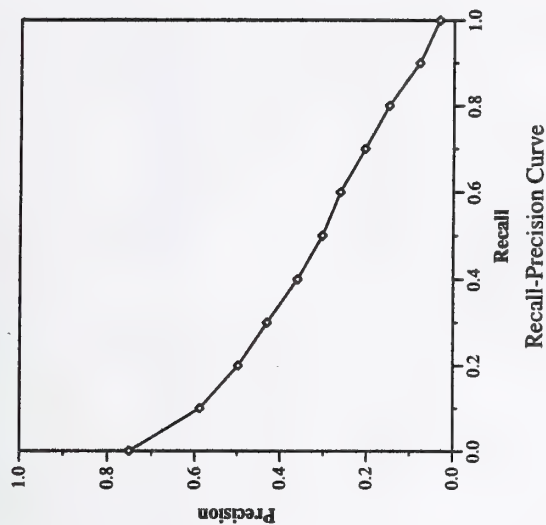
Document Level Averages	
	Precision
At 5 docs	0.5720
At 10 docs	0.5400
At 15 docs	0.4960
At 20 docs	0.4450
At 30 docs	0.4200
At 100 docs	0.2702
At 200 docs	0.1913
At 500 docs	0.1071
At 1000 docs	0.0634
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3381



Summary Statistics		
Run Number	att99atde	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3256	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7507
0.10	0.5886
0.20	0.4982
0.30	0.4307
0.40	0.3606
0.50	0.3032
0.60	0.2622
0.70	0.2047
0.80	0.1490
0.90	0.0791
1.00	0.0334
Average precision over all relevant docs	
non-interpolated	0.3165

Document Level Averages	
	Precision
At 5 docs	0.5640
At 10 docs	0.5480
At 15 docs	0.4933
At 20 docs	0.4680
At 30 docs	0.4253
At 100 docs	0.2758
At 200 docs	0.1974
At 500 docs	0.1100
At 1000 docs	0.0651
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3487

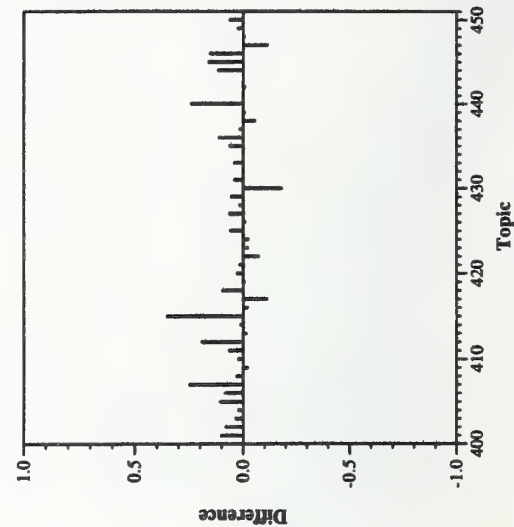
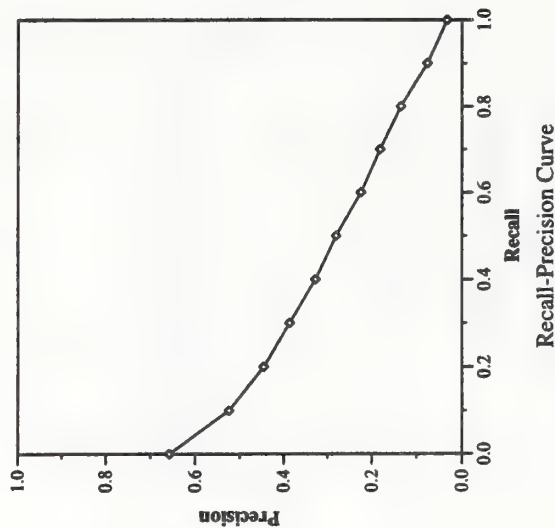


Ad hoc results — AT&T Labs Research

Summary Statistics		
Run Number	att99ate	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3140	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6578
0.10	0.5245
0.20	0.4462
0.30	0.3869
0.40	0.3277
0.50	0.2804
0.60	0.2247
0.70	0.1817
0.80	0.1355
0.90	0.0766
1.00	0.0331
Average precision over all relevant docs	
non-interpolated	0.2835

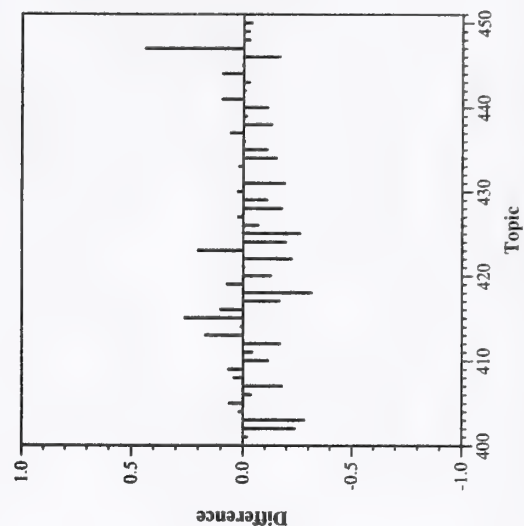
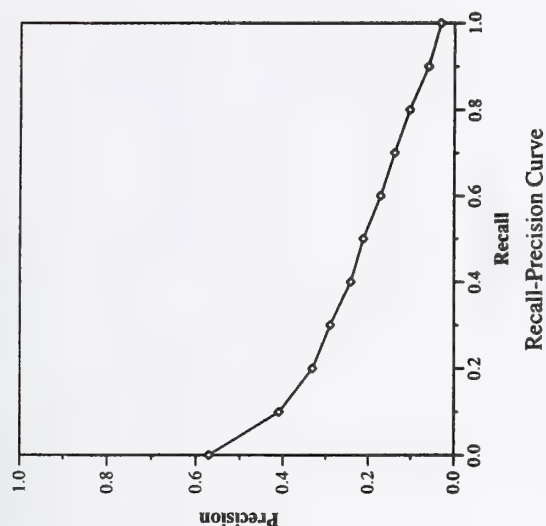
Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.4760
At 15 docs	0.4360
At 20 docs	0.4200
At 30 docs	0.3773
At 100 docs	0.2498
At 200 docs	0.1820
At 500 docs	0.1056
At 1000 docs	0.0628
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3154



Summary Statistics		
Run Number	weaver1	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2235	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5705
0.10	0.4092
0.20	0.3308
0.30	0.2897
0.40	0.2415
0.50	0.2118
0.60	0.1710
0.70	0.1386
0.80	0.1037
0.90	0.0601
1.00	0.0315
Average precision over all relevant docs	
non-interpolated	0.2175

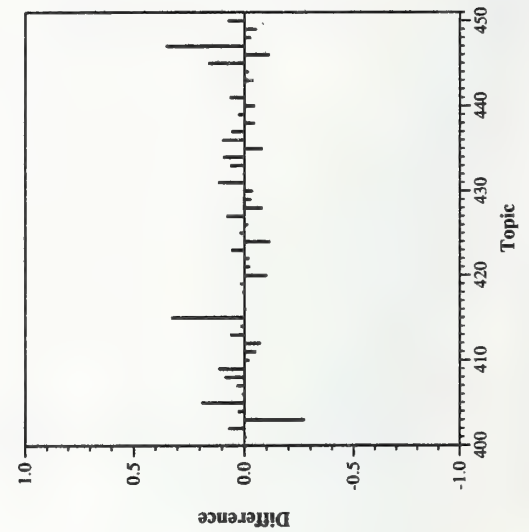
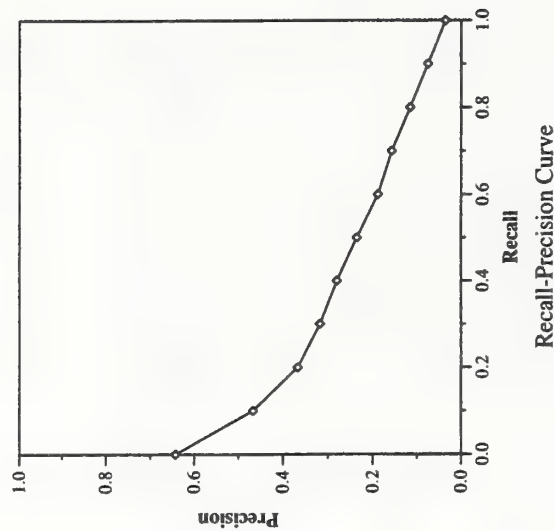
Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3500
At 15 docs	0.3000
At 20 docs	0.2880
At 30 docs	0.2547
At 100 docs	0.1686
At 200 docs	0.1197
At 500 docs	0.0700
At 1000 docs	0.0447
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2438



Summary Statistics		
Run Number	weaver2	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2942	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6426
0.10	0.4673
0.20	0.3671
0.30	0.3179
0.40	0.2804
0.50	0.2363
0.60	0.1887
0.70	0.1574
0.80	0.1157
0.90	0.0753
1.00	0.0355
Average precision over all relevant docs	
non-interpolated	0.2447

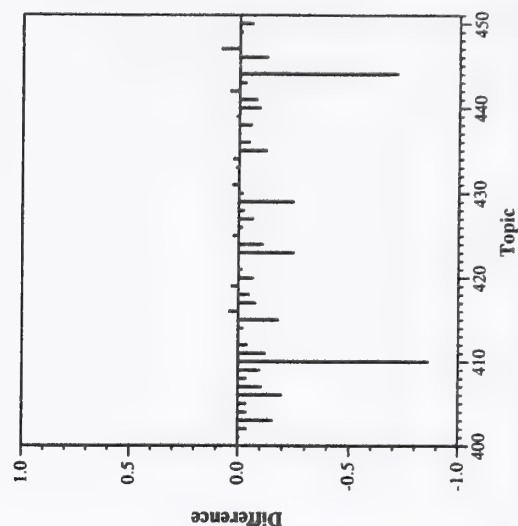
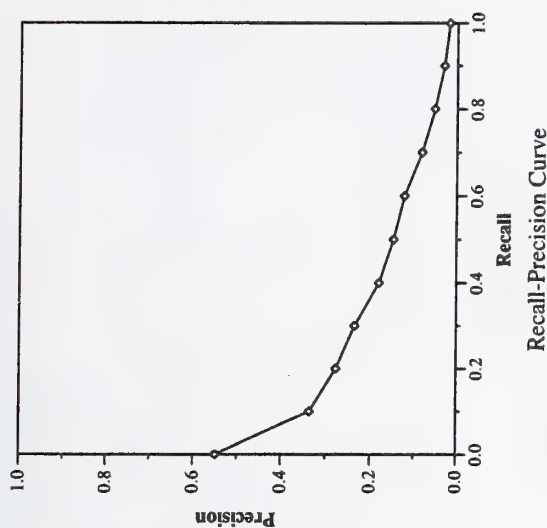
Document Level Averages	
	Precision
At 5 docs	0.4480
At 10 docs	0.4120
At 15 docs	0.3600
At 20 docs	0.3370
At 30 docs	0.3100
At 100 docs	0.2134
At 200 docs	0.1554
At 500 docs	0.0940
At 1000 docs	0.0588
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2696



Summary Statistics		
Run Number	plt8ahl	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	48259	
Relevant:	4728	
Rel-ret:	2444	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5487
0.10	0.3352
0.20	0.2768
0.30	0.2355
0.40	0.1805
0.50	0.1476
0.60	0.1228
0.70	0.0825
0.80	0.0530
0.90	0.0312
1.00	0.0191
Average precision over all relevant docs	
non-interpolated	0.1648

Document Level Averages	
	Precision
At 5 docs	0.3400
At 10 docs	0.3100
At 15 docs	0.3067
At 20 docs	0.2890
At 30 docs	0.2700
At 100 docs	0.1814
At 200 docs	0.1347
At 500 docs	0.0792
At 1000 docs	0.0489
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2164

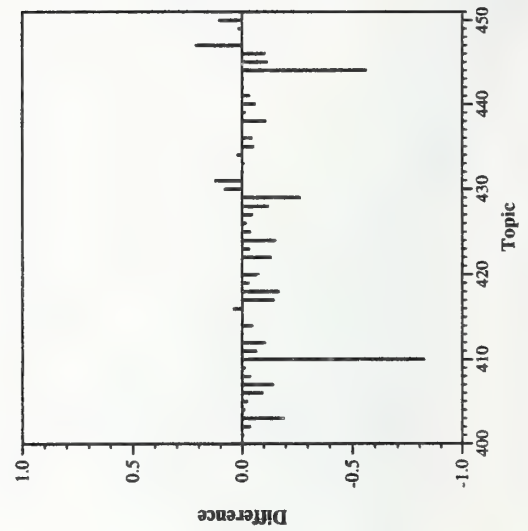
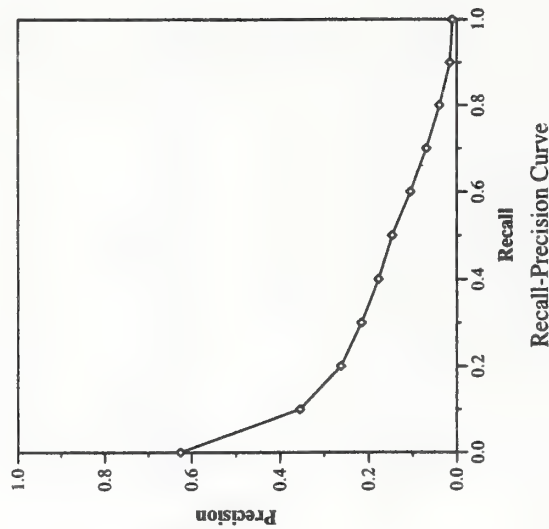


Ad hoc results — City University/Microsoft

Summary Statistics		
Run Number	plt8ah2	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	49998	
Relevant:	4728	
Rel-ret:	2240	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6258
0.10	0.3547
0.20	0.2618
0.30	0.2161
0.40	0.1776
0.50	0.1468
0.60	0.1053
0.70	0.0696
0.80	0.0396
0.90	0.0157
1.00	0.0101
Average precision over all relevant docs	
non-interpolated	0.1597

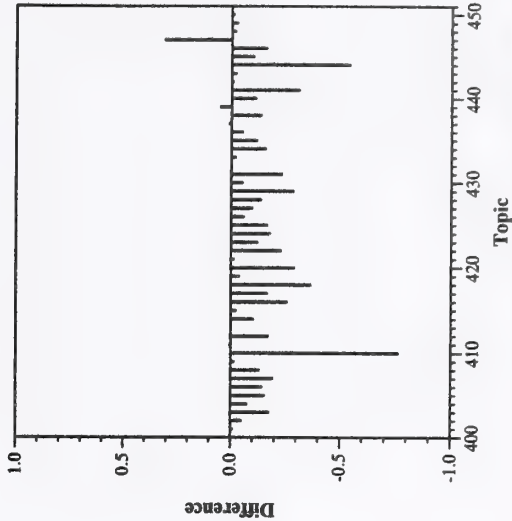
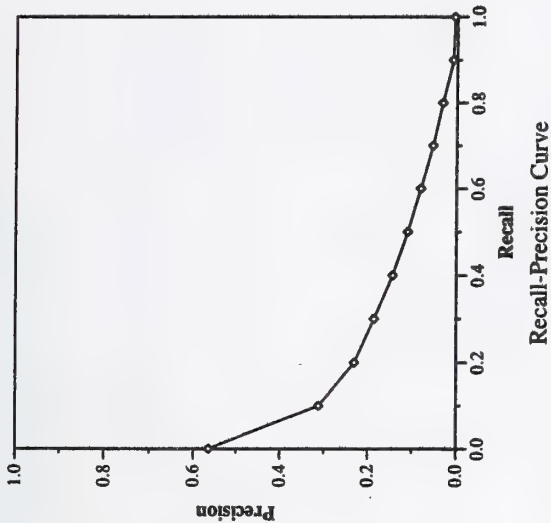
Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3320
At 15 docs	0.2947
At 20 docs	0.2670
At 30 docs	0.2307
At 100 docs	0.1498
At 200 docs	0.1134
At 500 docs	0.0688
At 1000 docs	0.0448
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2026



Summary Statistics		
Run Number	plt8ah3	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	1890	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5644
0.10	0.3117
0.20	0.2308
0.30	0.1861
0.40	0.1438
0.50	0.1098
0.60	0.0809
0.70	0.0535
0.80	0.0320
0.90	0.0087
1.00	0.0058
Average precision over all relevant docs	
non-interpolated	0.1353

Document Level Averages	
	Precision
At 5 docs	0.3480
At 10 docs	0.3100
At 15 docs	0.2693
At 20 docs	0.2410
At 30 docs	0.2147
At 100 docs	0.1352
At 200 docs	0.0961
At 500 docs	0.0580
At 1000 docs	0.0378
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1793

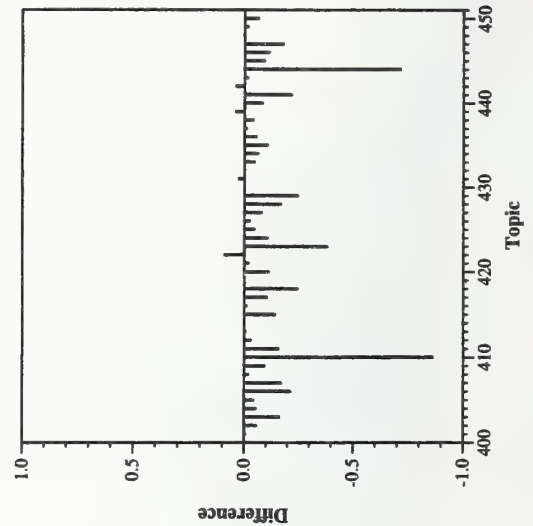
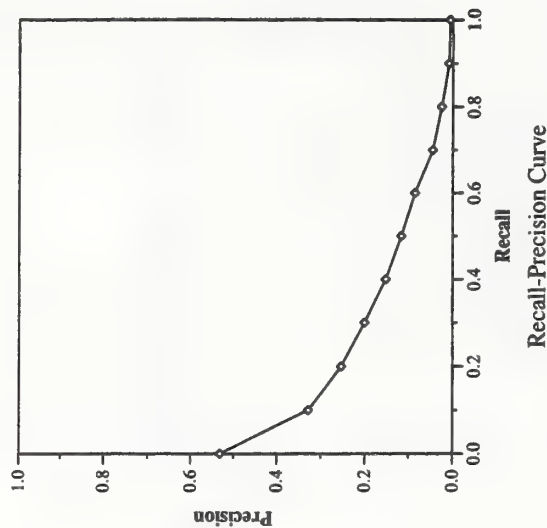


Ad hoc results — City University/Microsoft

Summary Statistics		
Run Number	plt8ah4	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	48256	
Relevant:	4728	
Rel-ret:	2215	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5318
0.10	0.3285
0.20	0.2539
0.30	0.2010
0.40	0.1523
0.50	0.1168
0.60	0.0861
0.70	0.0456
0.80	0.0255
0.90	0.0090
1.00	0.0066
Average precision over all relevant docs	
non-interpolated	0.1391

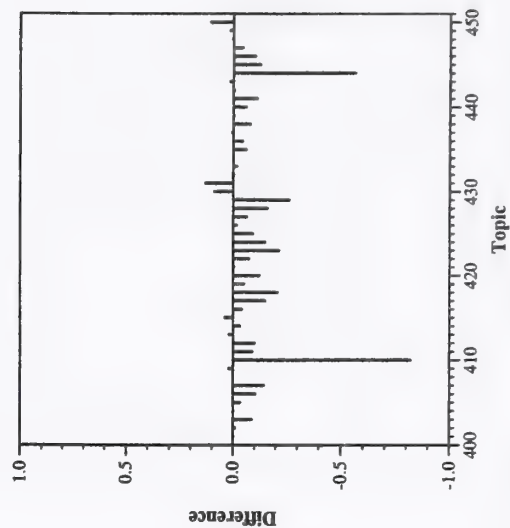
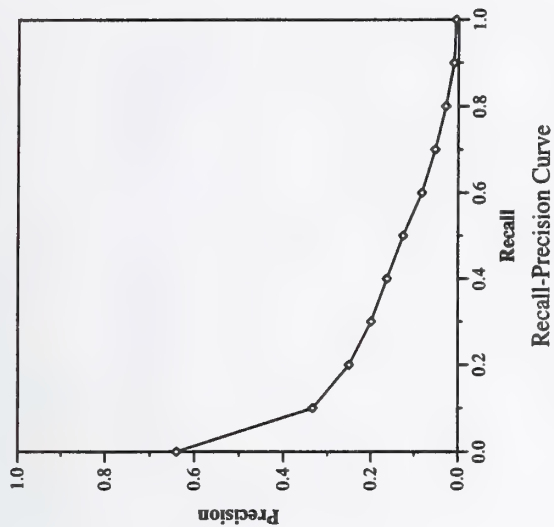
Document Level Averages	
	Precision
At 5 docs	0.3640
At 10 docs	0.3140
At 15 docs	0.2907
At 20 docs	0.2730
At 30 docs	0.2440
At 100 docs	0.1658
At 200 docs	0.1220
At 500 docs	0.0705
At 1000 docs	0.0443
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1981



Summary Statistics		
Run Number	plt8ah5	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	49997	
Relevant:	4728	
Rel-ret:	2195	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6400
0.10	0.3336
0.20	0.2504
0.30	0.1992
0.40	0.1628
0.50	0.1258
0.60	0.0827
0.70	0.0527
0.80	0.0284
0.90	0.0102
1.00	0.0060
Average precision over all relevant docs	
non-interpolated	0.1493

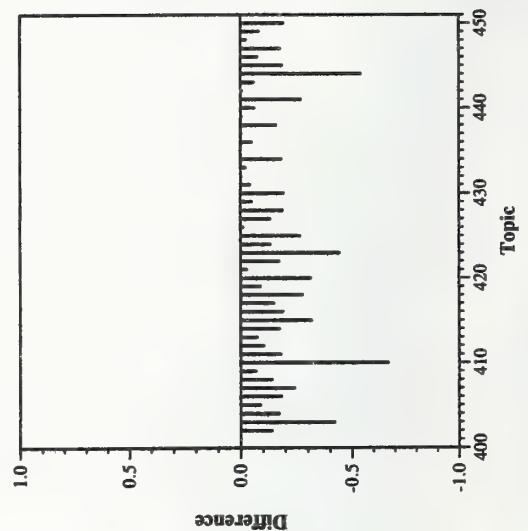
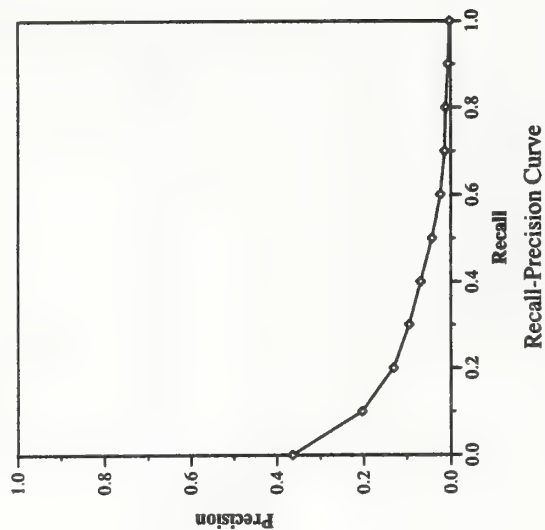
Document Level Averages	
	Precision
At 5 docs	0.3880
At 10 docs	0.3220
At 15 docs	0.2853
At 20 docs	0.2590
At 30 docs	0.2240
At 100 docs	0.1428
At 200 docs	0.1101
At 500 docs	0.0686
At 1000 docs	0.0439
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1876



Summary Statistics		
Run Number	Dm8NbnR	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	1688	

Recall Level Precision Averages	
Recall	Precision
0.00	0.3641
0.10	0.2038
0.20	0.1309
0.30	0.0950
0.40	0.0690
0.50	0.0428
0.60	0.0238
0.70	0.0140
0.80	0.0116
0.90	0.0055
1.00	0.0023
Average precision over all relevant docs	
non-interpolated	0.0712

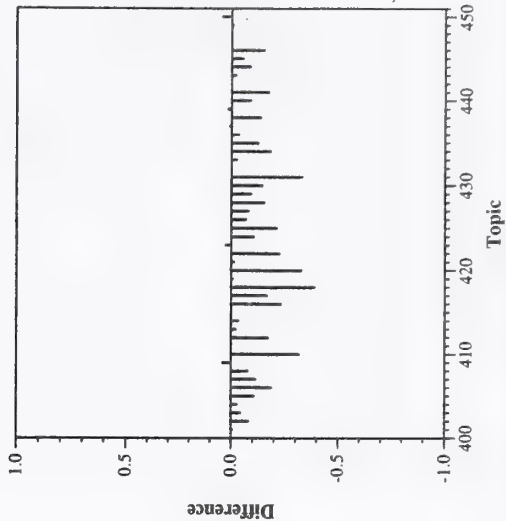
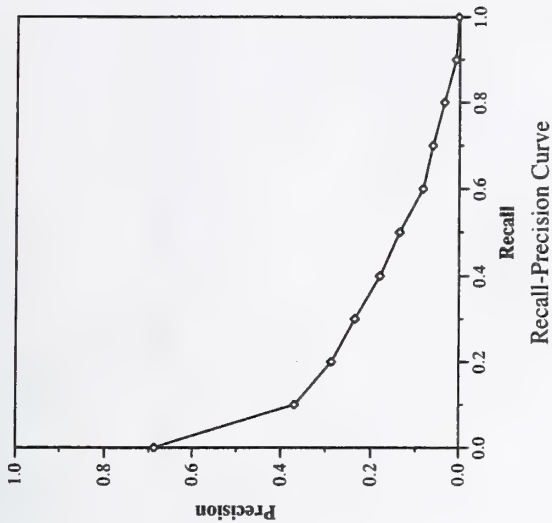
Document Level Averages	
	Precision
At 5 docs	0.2120
At 10 docs	0.1880
At 15 docs	0.1600
At 20 docs	0.1490
At 30 docs	0.1333
At 100 docs	0.0908
At 200 docs	0.0695
At 500 docs	0.0475
At 1000 docs	0.0338
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1241



Summary Statistics		
Run Number	Dm8TFidf	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	1922	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6879
0.10	0.3693
0.20	0.2870
0.30	0.2356
0.40	0.1805
0.50	0.1366
0.60	0.0841
0.70	0.0620
0.80	0.0353
0.90	0.0084
1.00	0.0026
Average precision over all relevant docs	
non-interpolated	0.1630

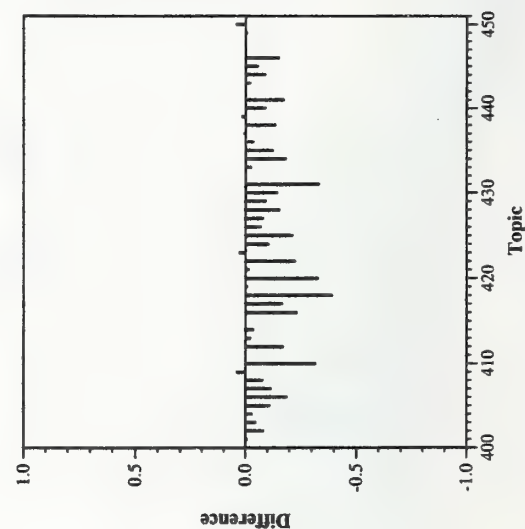
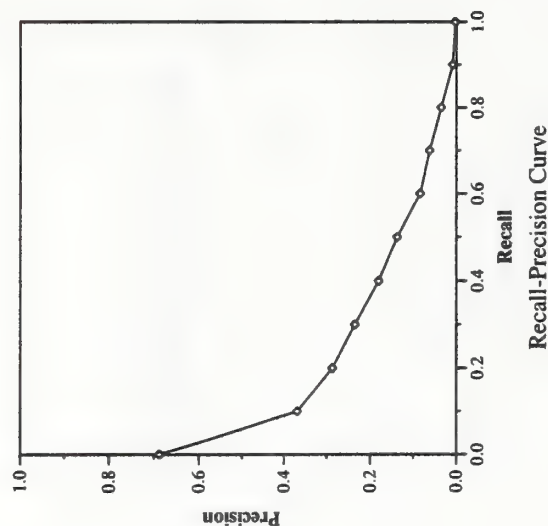
Document Level Averages	
	Precision
At 5 docs	0.4040
At 10 docs	0.3440
At 15 docs	0.3120
At 20 docs	0.2850
At 30 docs	0.2513
At 100 docs	0.1414
At 200 docs	0.0995
At 500 docs	0.0604
At 1000 docs	0.0384
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2131



Summary Statistics		
Run Number	Dm8TFbn	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	1922	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6879
0.10	0.3693
0.20	0.2870
0.30	0.2356
0.40	0.1805
0.50	0.1374
0.60	0.0841
0.70	0.0619
0.80	0.0353
0.90	0.0083
1.00	0.0026
Average precision over all relevant docs	
non-interpolated	0.1630

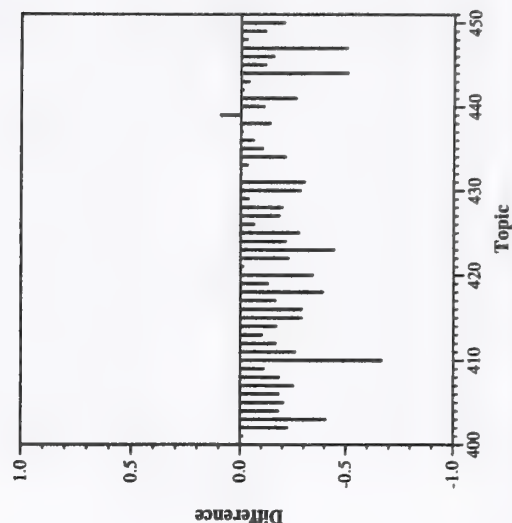
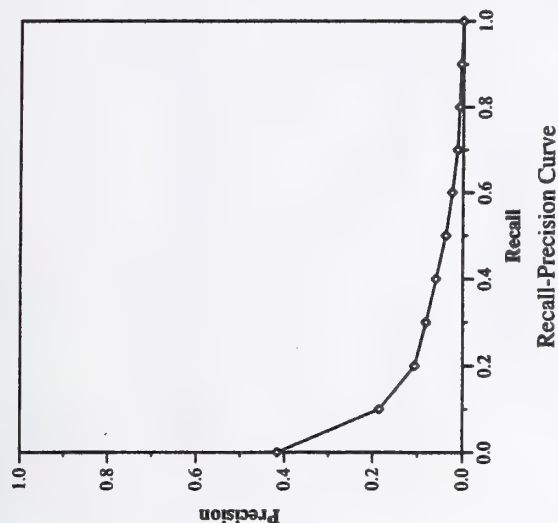
Document Level Averages	
	Precision
At 5 docs	0.4040
At 10 docs	0.3440
At 15 docs	0.3133
At 20 docs	0.2850
At 30 docs	0.2507
At 100 docs	0.1414
At 200 docs	0.0995
At 500 docs	0.0604
At 1000 docs	0.0384
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2131



Summary Statistics		
Run Number	Dm8Nbn	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	1510	

Recall Level Precision Averages	
Recall	Precision
0.00	0.4158
0.10	0.1853
0.20	0.1066
0.30	0.0814
0.40	0.0589
0.50	0.0375
0.60	0.0239
0.70	0.0126
0.80	0.0082
0.90	0.0048
1.00	0.0006
Average precision over all relevant docs	
non-interpolated	0.0675

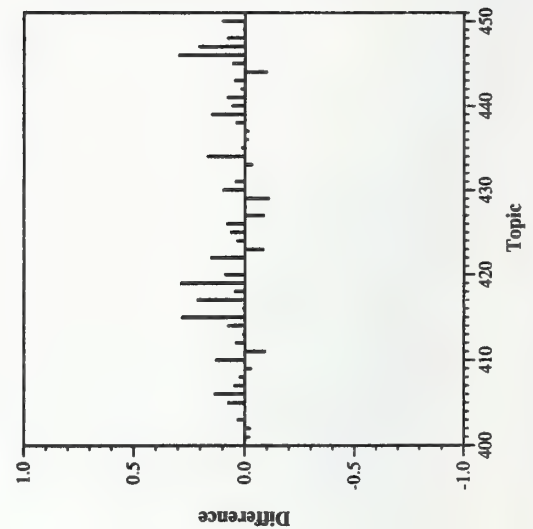
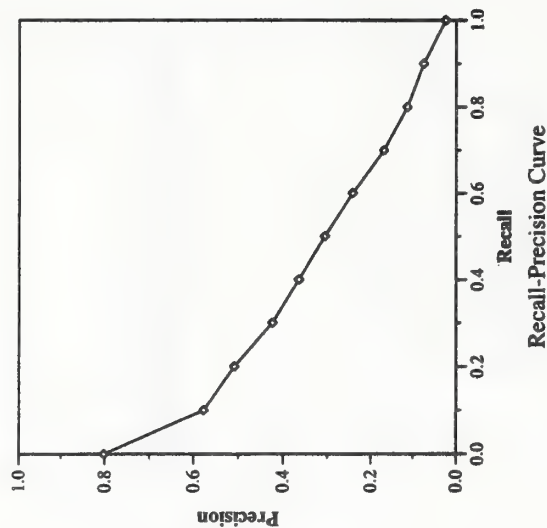
Document Level Averages	
	Precision
At 5 docs	0.2320
At 10 docs	0.1900
At 15 docs	0.1680
At 20 docs	0.1530
At 30 docs	0.1367
At 100 docs	0.0820
At 200 docs	0.0618
At 500 docs	0.0430
At 1000 docs	0.0302
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1130



Summary Statistics		
Run Number	fub99tf	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics.		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3281	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8019
0.10	0.5774
0.20	0.5090
0.30	0.4242
0.40	0.3653
0.50	0.3062
0.60	0.2424
0.70	0.1696
0.80	0.1137
0.90	0.0753
1.00	0.0242
Average precision over all relevant docs	
non-interpolated	0.3099

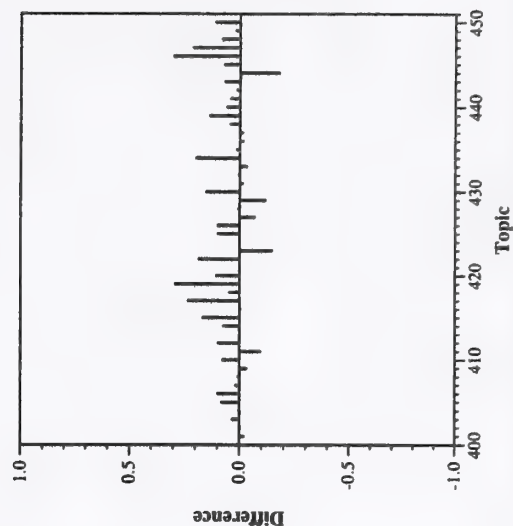
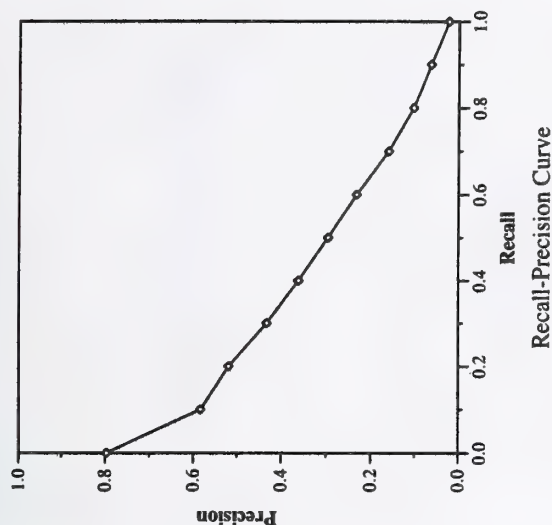
Document Level Averages	
	Precision
At 5 docs	0.6040
At 10 docs	0.5260
At 15 docs	0.4760
At 20 docs	0.4360
At 30 docs	0.3973
At 100 docs	0.2596
At 200 docs	0.1904
At 500 docs	0.1088
At 1000 docs	0.0656
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3398



Summary Statistics		
Run Number	fub99a	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3262	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7968
0.10	0.5833
0.20	0.5189
0.30	0.4322
0.40	0.3616
0.50	0.2963
0.60	0.2322
0.70	0.1600
0.80	0.1027
0.90	0.0631
1.00	0.0228
Average precision over all relevant docs	
non-interpolated	0.3068

Document Level Averages	
	Precision
At 5 docs	0.6080
At 10 docs	0.5300
At 15 docs	0.4827
At 20 docs	0.4430
At 30 docs	0.4053
At 100 docs	0.2620
At 200 docs	0.1898
At 500 docs	0.1088
At 1000 docs	0.0652
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3364

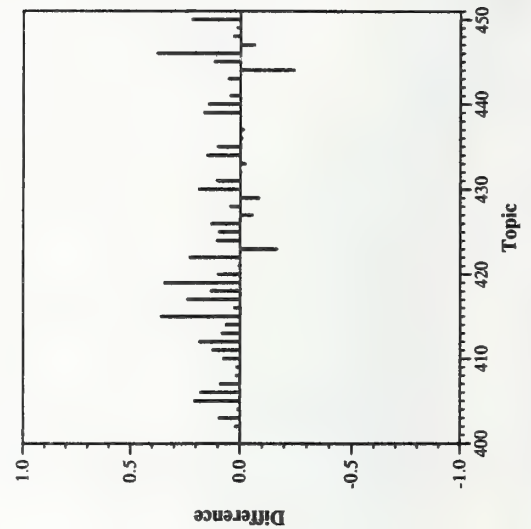
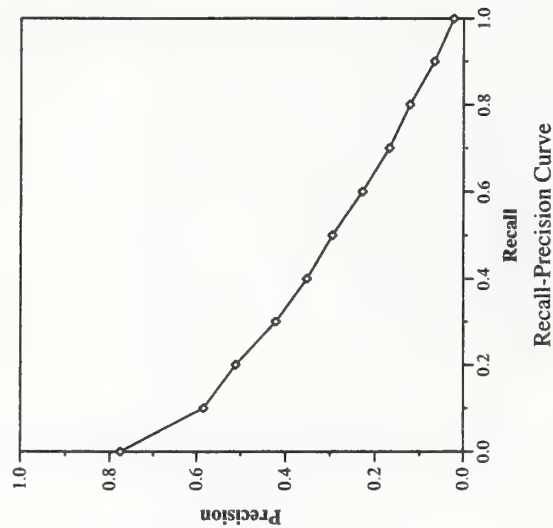


Ad hoc results — Fondazione Ugo Bordoni

Summary Statistics		
Run Number	fub99td	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3298	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7763
0.10	0.5845
0.20	0.5115
0.30	0.4217
0.40	0.3522
0.50	0.2952
0.60	0.2290
0.70	0.1689
0.80	0.1229
0.90	0.0671
1.00	0.0226
Average precision over all relevant docs	
non-interpolated	0.3064

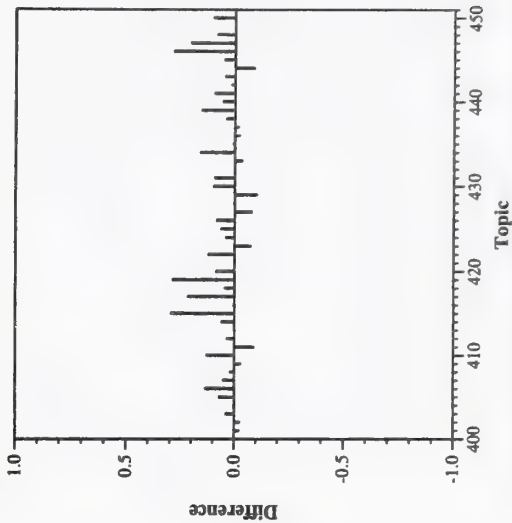
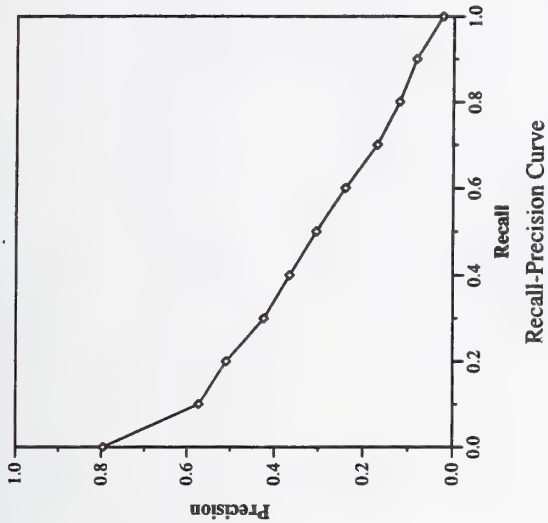
Document Level Averages	
	Precision
At 5 docs	0.5760
At 10 docs	0.5100
At 15 docs	0.4667
At 20 docs	0.4470
At 30 docs	0.4013
At 100 docs	0.2772
At 200 docs	0.1981
At 500 docs	0.1107
At 1000 docs	0.0660
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3366



Summary Statistics		
Run Number	fub99tt	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3299	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7962
0.10	0.5720
0.20	0.5093
0.30	0.4246
0.40	0.3663
0.50	0.3057
0.60	0.2410
0.70	0.1699
0.80	0.1205
0.90	0.0830
1.00	0.0245
Average precision over all relevant docs	
non-interpolated	0.3106

Document Level Averages	
	Precision
At 5 docs	0.6160
At 10 docs	0.5300
At 15 docs	0.4747
At 20 docs	0.4380
At 30 docs	0.3973
At 100 docs	0.2574
At 200 docs	0.1893
At 500 docs	0.1092
At 1000 docs	0.0660
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3354

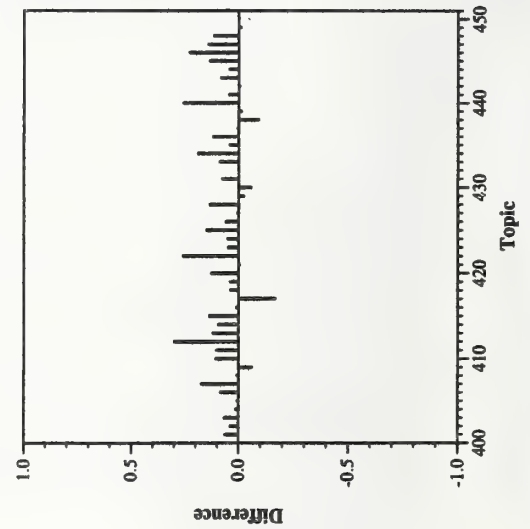
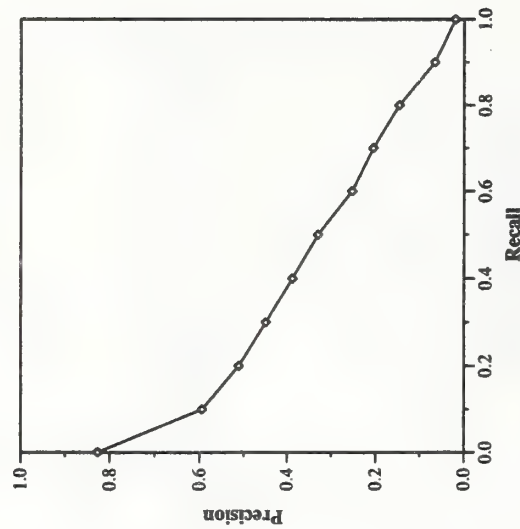


Ad hoc results — Fujitsu Laboratories. Ltd.

Summary Statistics		
Run Number	Flab8atdn	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3261	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8269
0.10	0.5945
0.20	0.5119
0.30	0.4491
0.40	0.3876
0.50	0.3299
0.60	0.2535
0.70	0.2054
0.80	0.1462
0.90	0.0657
1.00	0.0196
Average precision over all relevant docs	
non-interpolated	0.3240

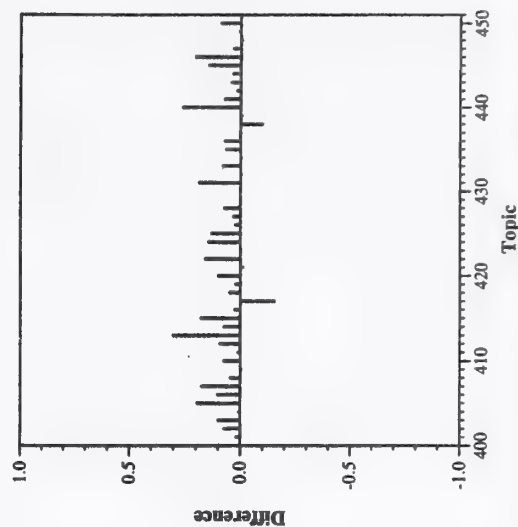
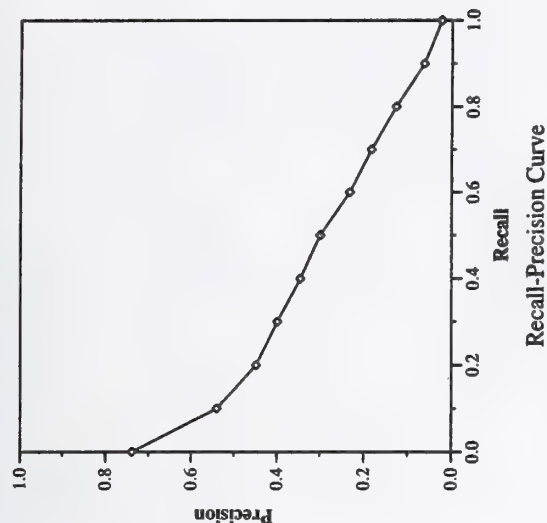
Document Level Averages	
	Precision
At 5 docs	0.6040
At 10 docs	0.5460
At 15 docs	0.5053
At 20 docs	0.4700
At 30 docs	0.4240
At 100 docs	0.2800
At 200 docs	0.1965
At 500 docs	0.1097
At 1000 docs	0.0652
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3538



Summary Statistics		
Run Number	Flab8as	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3090	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7387
0.10	0.5402
0.20	0.4488
0.30	0.3989
0.40	0.3451
0.50	0.2990
0.60	0.2327
0.70	0.1834
0.80	0.1272
0.90	0.0635
1.00	0.0237
Average precision over all relevant docs	
non-interpolated	0.2901

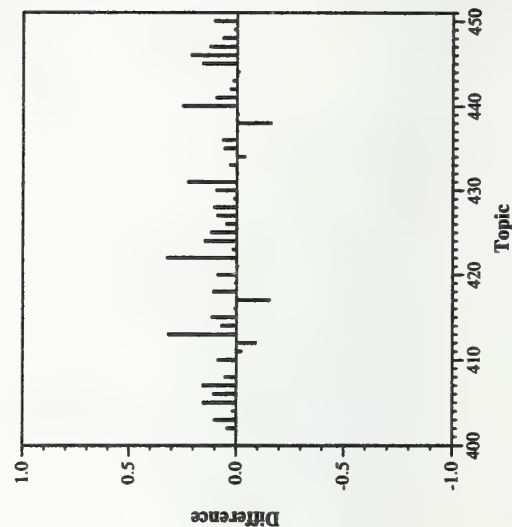
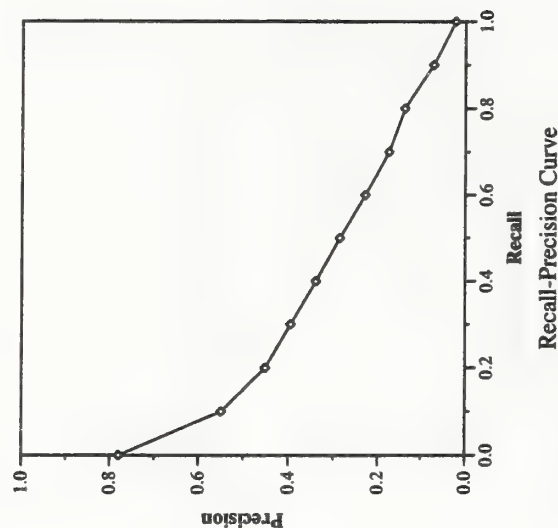
Document Level Averages	
	Precision
At 5 docs	0.5320
At 10 docs	0.4860
At 15 docs	0.4507
At 20 docs	0.4200
At 30 docs	0.3867
At 100 docs	0.2546
At 200 docs	0.1832
At 500 docs	0.1052
At 1000 docs	0.0618
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3248



Summary Statistics		
Run Number	Flab8atd2	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2990	

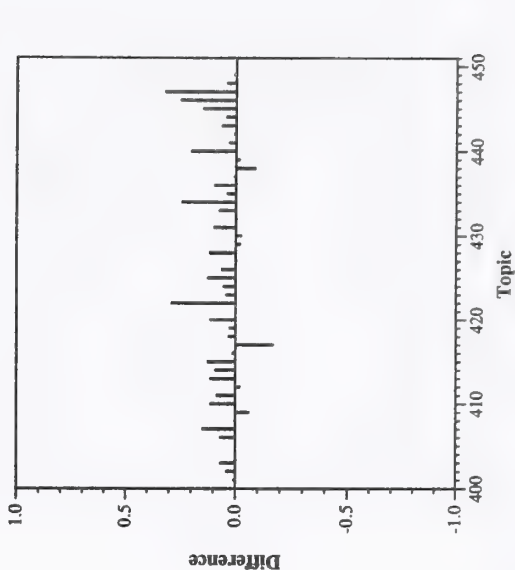
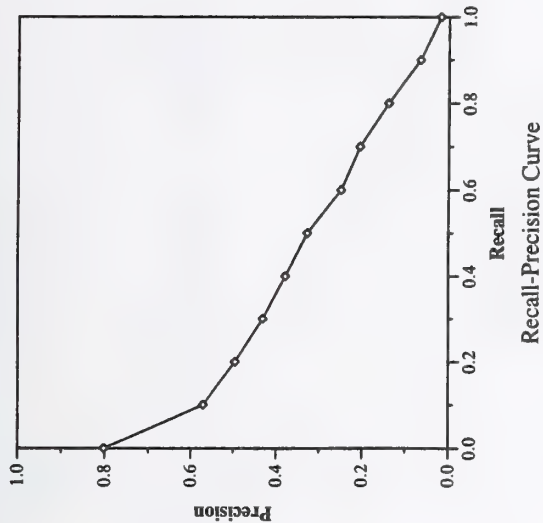
Recall Level Precision Averages	
Recall	Precision
0.00	0.7796
0.10	0.5490
0.20	0.4517
0.30	0.3954
0.40	0.3397
0.50	0.2863
0.60	0.2291
0.70	0.1745
0.80	0.1381
0.90	0.0720
1.00	0.0224
Average precision over all relevant docs	
non-interpolated	0.2930

Document Level Averages	
	Precision
At 5 docs	0.5480
At 10 docs	0.4880
At 15 docs	0.4587
At 20 docs	0.4200
At 30 docs	0.3887
At 100 docs	0.2490
At 200 docs	0.1777
At 500 docs	0.1011
At 1000 docs	0.0598
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3203



Summary Statistics		
Run Number	Flab8ax	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3207	

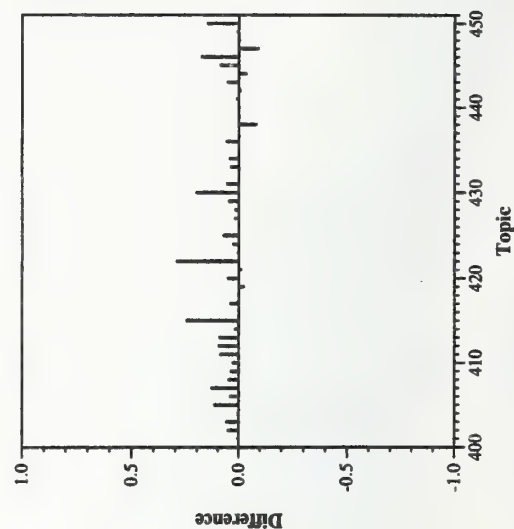
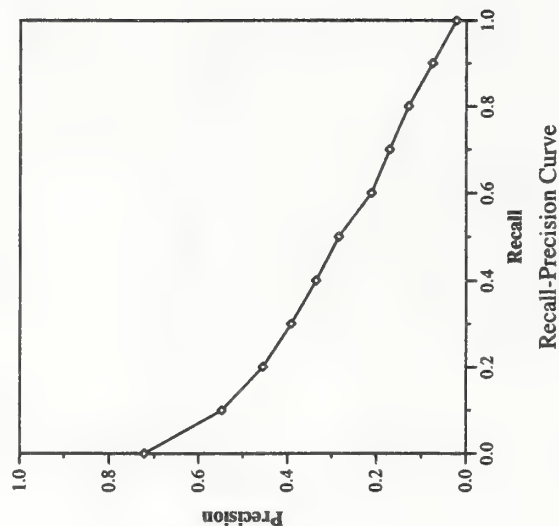
Recall Level Precision Averages		Document Level Averages	
Recall	Precision		Precision
0.00	0.8020	At 5 docs	0.5840
0.10	0.5710	At 10 docs	0.5240
0.20	0.4960	At 15 docs	0.4773
0.30	0.4317	At 20 docs	0.4510
0.40	0.3790	At 30 docs	0.4027
0.50	0.3280	At 100 docs	0.2690
0.60	0.2493	At 200 docs	0.1916
0.70	0.2055	At 500 docs	0.1089
0.80	0.1394	At 1000 docs	0.0641
0.90	0.0658	R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
1.00	0.0185		
Average precision over all relevant docs			
non-interpolated	0.3163	Exact	0.3508



Summary Statistics		
Run Number	Flab8at	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3084	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7227
0.10	0.5471
0.20	0.4550
0.30	0.3913
0.40	0.3352
0.50	0.2840
0.60	0.2112
0.70	0.1709
0.80	0.1289
0.90	0.0757
1.00	0.0221
Average precision over all relevant docs	
non-interpolated	0.2876

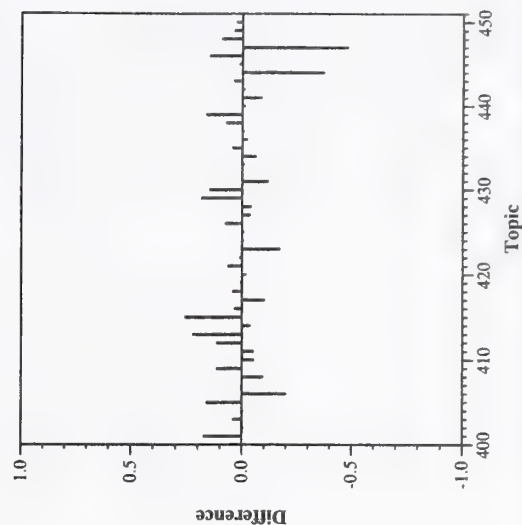
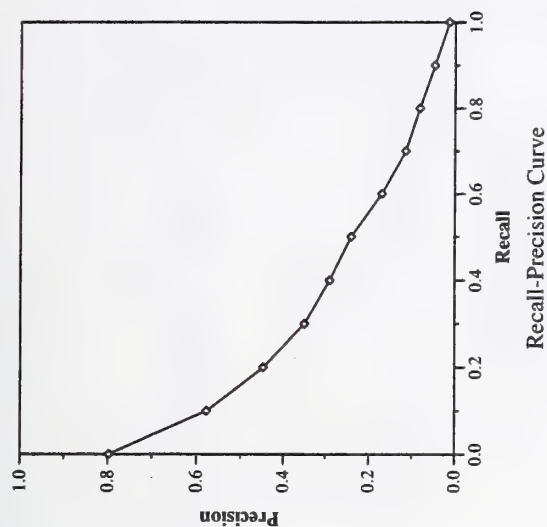
Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.4860
At 15 docs	0.4533
At 20 docs	0.4260
At 30 docs	0.3907
At 100 docs	0.2554
At 200 docs	0.1795
At 500 docs	0.1036
At 1000 docs	0.0617
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3153



Summary Statistics	
Run Number	GE8ATDN1
Run Description	Automatic, title + desc + narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3138

Recall Level Precision Averages	
Recall	Precision
0.00	0.7976
0.10	0.5776
0.20	0.4476
0.30	0.3499
0.40	0.2901
0.50	0.2405
0.60	0.1701
0.70	0.1152
0.80	0.0829
0.90	0.0481
1.00	0.0136
Average precision over all relevant docs	
non-interpolated	0.2623

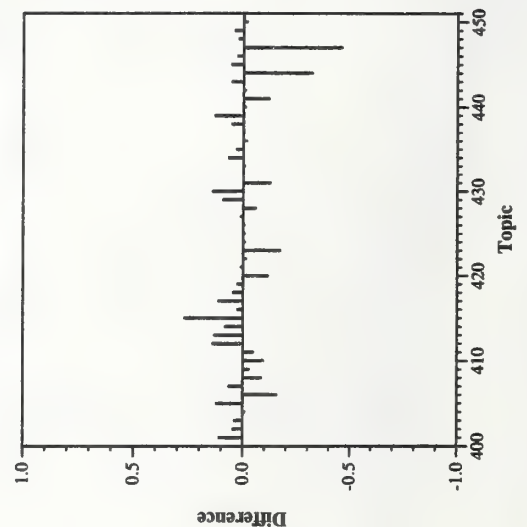
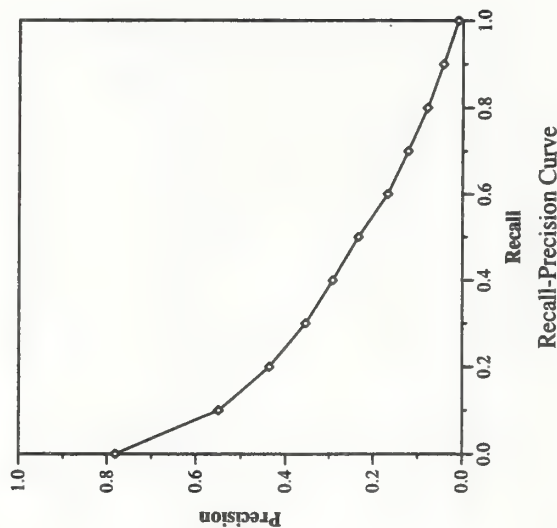
Document Level Averages	
	Precision
At 5 docs	0.5560
At 10 docs	0.5020
At 15 docs	0.4693
At 20 docs	0.4330
At 30 docs	0.3900
At 100 docs	0.2460
At 200 docs	0.1777
At 500 docs	0.1023
At 1000 docs	0.0628
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2984



Summary Statistics		
Run Number	GE8ATDN2	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3068	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7815
0.10	0.5498
0.20	0.4370
0.30	0.3547
0.40	0.2924
0.50	0.2341
0.60	0.1683
0.70	0.1227
0.80	0.0795
0.90	0.0436
1.00	0.0105
Average precision over all relevant docs	
non-interpolated	0.2580

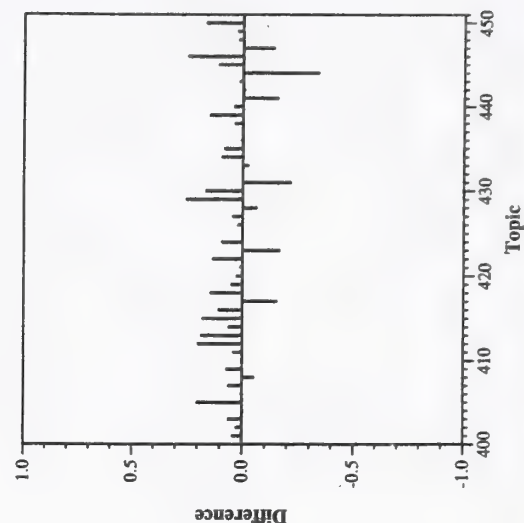
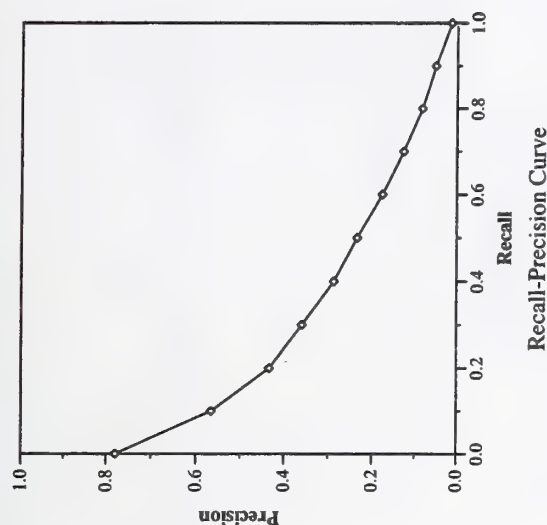
Document Level Averages	
	Precision
At 5 docs	0.5480
At 10 docs	0.5120
At 15 docs	0.4627
At 20 docs	0.4300
At 30 docs	0.3860
At 100 docs	0.2492
At 200 docs	0.1783
At 500 docs	0.1017
At 1000 docs	0.0614
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2993



Summary Statistics	
Run Number	GE8ATD3
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3022

Recall Level Precision Averages	
Recall	Precision
0.00	0.7795
0.10	0.5658
0.20	0.4334
0.30	0.3596
0.40	0.2873
0.50	0.2348
0.60	0.1766
0.70	0.1274
0.80	0.0834
0.90	0.0514
1.00	0.0137
Average precision over all relevant docs	
non-interpolated	0.2618

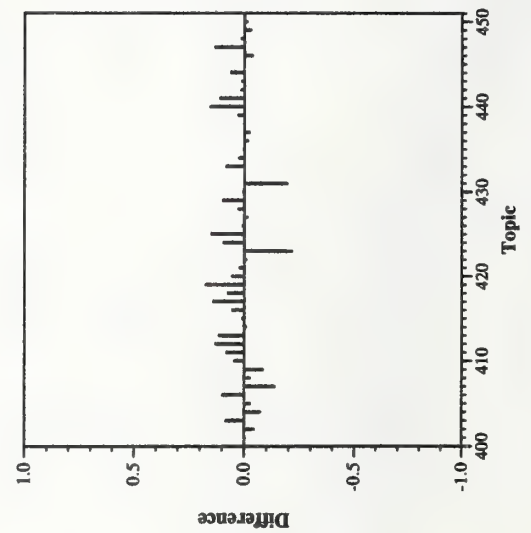
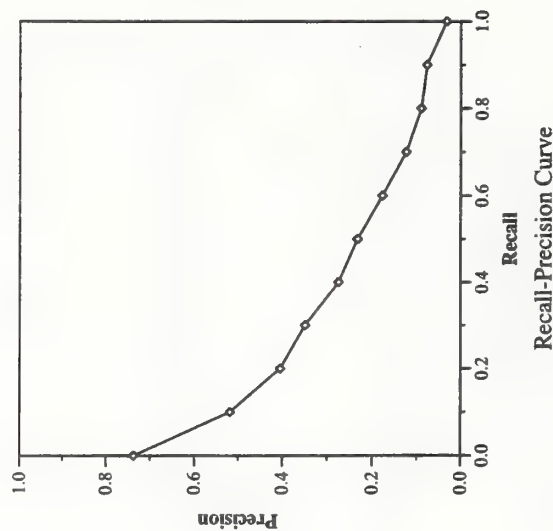
Document Level Averages	
	Precision
At 5 docs	0.5280
At 10 docs	0.4820
At 15 docs	0.4507
At 20 docs	0.4250
At 30 docs	0.3813
At 100 docs	0.2404
At 200 docs	0.1734
At 500 docs	0.1000
At 1000 docs	0.0604
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2959



Summary Statistics		
Run Number	ibmg99a	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2807	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7372
0.10	0.5183
0.20	0.4042
0.30	0.3488
0.40	0.2750
0.50	0.2327
0.60	0.1772
0.70	0.1240
0.80	0.0898
0.90	0.0768
1.00	0.0319
Average precision over all relevant docs	
non-interpolated	0.2484

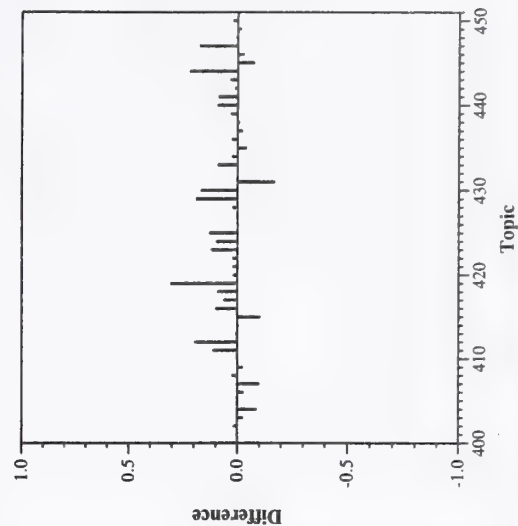
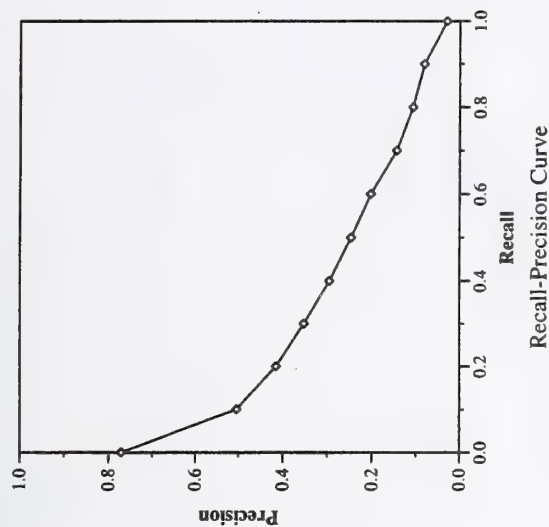
Document Level Averages	
	Precision
At 5 docs	0.4720
At 10 docs	0.4200
At 15 docs	0.3867
At 20 docs	0.3640
At 30 docs	0.3333
At 100 docs	0.2266
At 200 docs	0.1578
At 500 docs	0.0936
At 1000 docs	0.0561
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2866



Summary Statistics		
Run Number	ibmg99b	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2807	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7713
0.10	0.5044
0.20	0.4154
0.30	0.3526
0.40	0.2952
0.50	0.2463
0.60	0.2025
0.70	0.1441
0.80	0.1074
0.90	0.0814
1.00	0.0295
Average precision over all relevant docs	
non-interpolated	0.2609

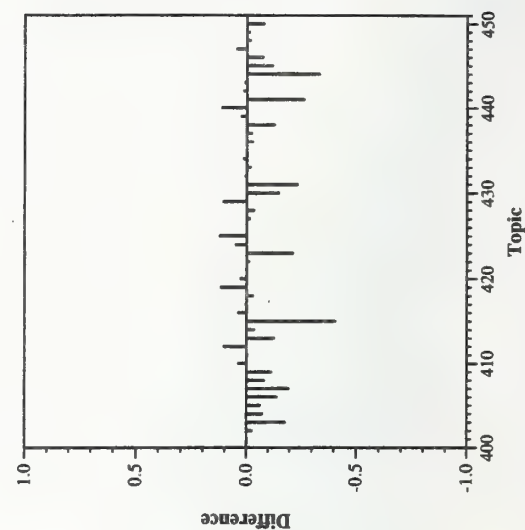
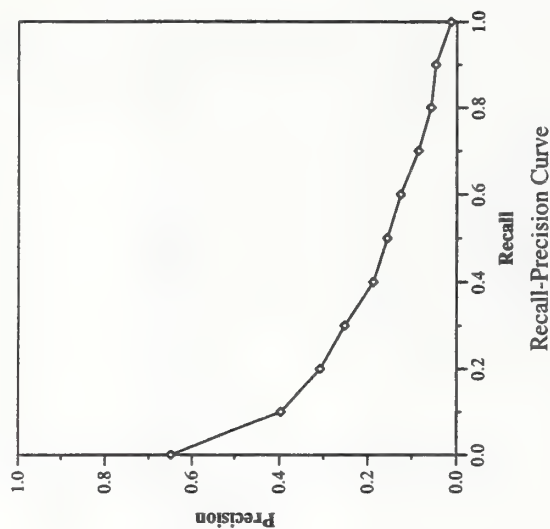
Document Level Averages	
	Precision
At 5 docs	0.4840
At 10 docs	0.4440
At 15 docs	0.4200
At 20 docs	0.3980
At 30 docs	0.3573
At 100 docs	0.2270
At 200 docs	0.1604
At 500 docs	0.0927
At 1000 docs	0.0561
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3128



Summary Statistics		
Run Number	ibmg99c	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	42765	
Relevant:	4728	
Rel-ret:	2461	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6484
0.10	0.3967
0.20	0.3078
0.30	0.2522
0.40	0.1877
0.50	0.1560
0.60	0.1259
0.70	0.0855
0.80	0.0579
0.90	0.0474
1.00	0.0133
Average precision over all relevant docs	
non-interpolated	0.1782

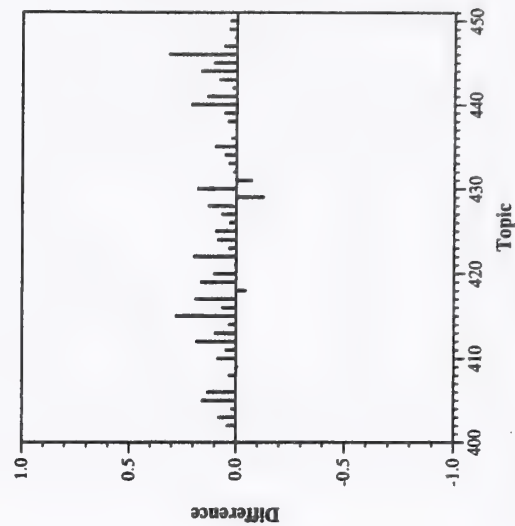
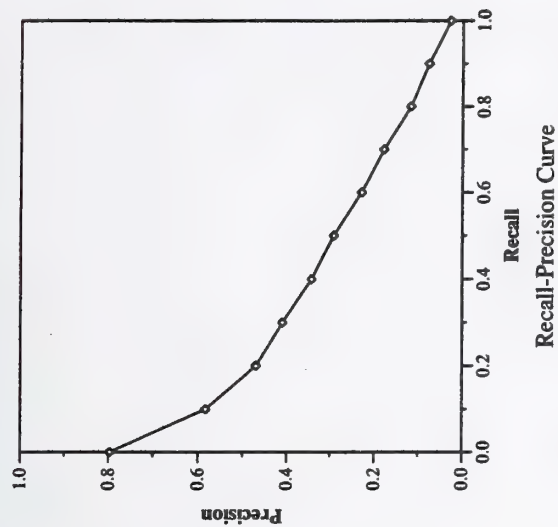
Document Level Averages	
	Precision
At 5 docs	0.3440
At 10 docs	0.3040
At 15 docs	0.2640
At 20 docs	0.2450
At 30 docs	0.2260
At 100 docs	0.1842
At 200 docs	0.1366
At 500 docs	0.0825
At 1000 docs	0.0492
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2329



Summary Statistics		
Run Number	ibms99a	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3195	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7966
0.10	0.5834
0.20	0.4698
0.30	0.4095
0.40	0.3429
0.50	0.2918
0.60	0.2289
0.70	0.1783
0.80	0.1173
0.90	0.0764
1.00	0.0278
Average precision over all relevant docs	
non-interpolated	0.3005

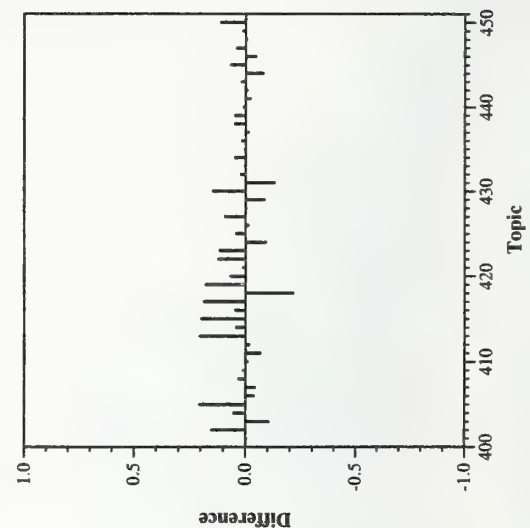
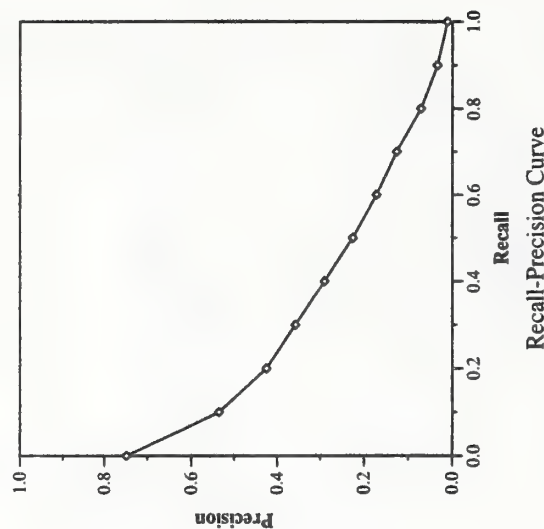
Document Level Averages	
At 5 docs	0.5880
At 10 docs	0.5040
At 15 docs	0.4720
At 20 docs	0.4500
At 30 docs	0.4100
At 100 docs	0.2614
At 200 docs	0.1870
At 500 docs	0.1058
At 1000 docs	0.0639
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3362



Summary Statistics	
Run Number	ibms99c
Run Description	Automatic, desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	2836

Recall Level Precision Averages	
Recall	Precision
0.00	0.7479
0.10	0.5350
0.20	0.4250
0.30	0.3578
0.40	0.2898
0.50	0.2251
0.60	0.1719
0.70	0.1258
0.80	0.0711
0.90	0.0342
1.00	0.0115
Average precision over all relevant docs	
non-interpolated	0.2531

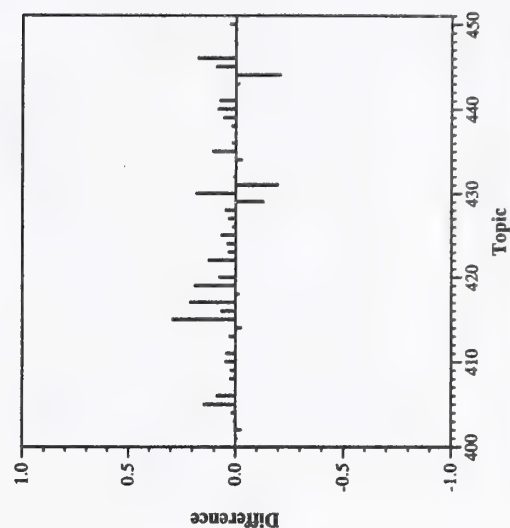
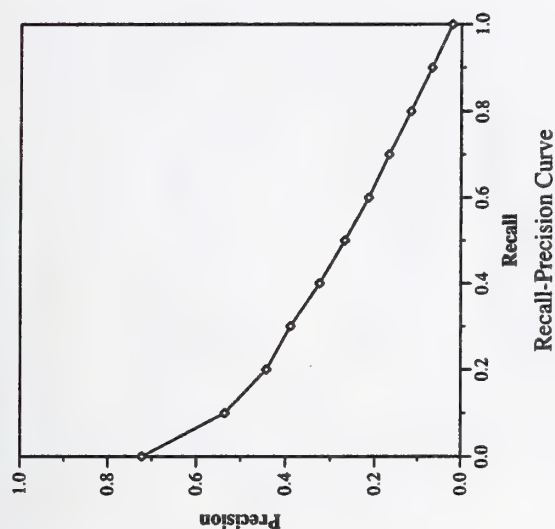
Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.4700
At 15 docs	0.4387
At 20 docs	0.3950
At 30 docs	0.3453
At 100 docs	0.2186
At 200 docs	0.1579
At 500 docs	0.0924
At 1000 docs	0.0567
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2925



Summary Statistics		
Run Number	ibms99b	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3151	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7224
0.10	0.5354
0.20	0.4424
0.30	0.3888
0.40	0.3241
0.50	0.2682
0.60	0.2148
0.70	0.1678
0.80	0.1168
0.90	0.0676
1.00	0.0207
Average precision over all relevant docs	
non-interpolated	0.2784

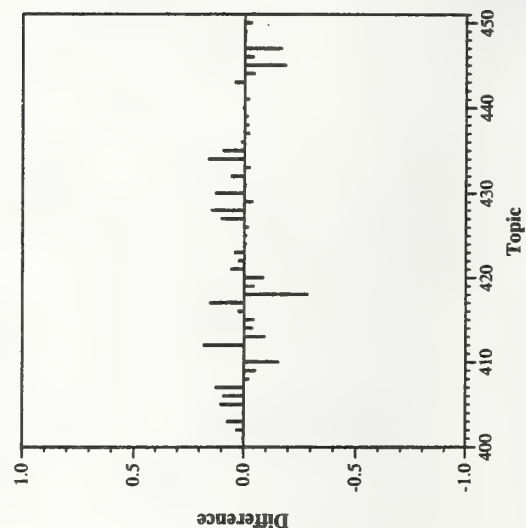
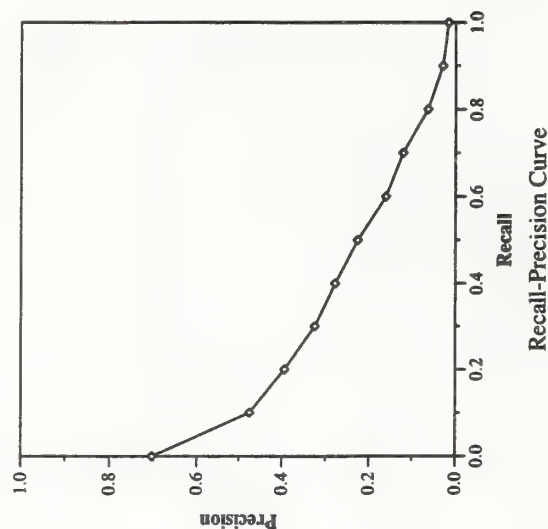
Document Level Averages	
	Precision
At 5 docs	0.5000
At 10 docs	0.4600
At 15 docs	0.4227
At 20 docs	0.4090
At 30 docs	0.3787
At 100 docs	0.2494
At 200 docs	0.1802
At 500 docs	0.1025
At 1000 docs	0.0630
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3190



Summary Statistics	
Run Number	iit99aul
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	48766
Relevant:	4728
Rel-ret:	2688

Recall Level Precision Averages	
Recall	Precision
0.00	0.7039
0.10	0.4749
0.20	0.3941
0.30	0.3241
0.40	0.2775
0.50	0.2255
0.60	0.1603
0.70	0.1210
0.80	0.0637
0.90	0.0296
1.00	0.0169
Average precision over all relevant docs	
non-interpolated	0.2305

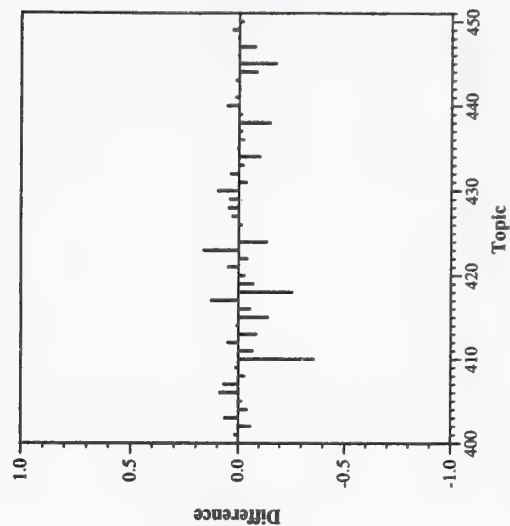
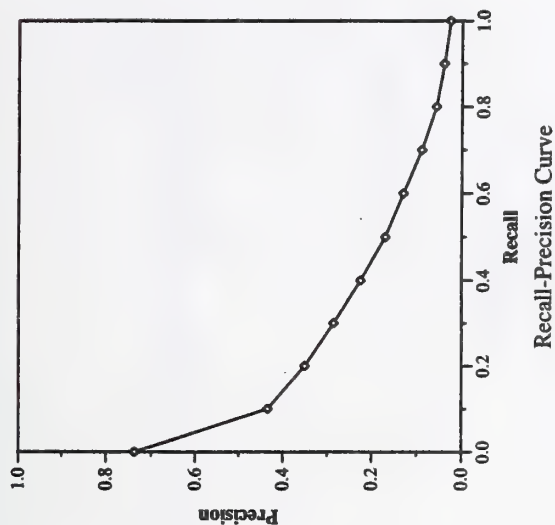
Document Level Averages	
	Precision
At 5 docs	0.4920
At 10 docs	0.4320
At 15 docs	0.4000
At 20 docs	0.3760
At 30 docs	0.3360
At 100 docs	0.2146
At 200 docs	0.1512
At 500 docs	0.0864
At 1000 docs	0.0538
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2731



Summary Statistics		
Run Number	iit99au2	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	47813	
Relevant:	4728	
Rel-ret:	2207	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7366
0.10	0.4343
0.20	0.3509
0.30	0.2854
0.40	0.2251
0.50	0.1695
0.60	0.1293
0.70	0.0887
0.80	0.0569
0.90	0.0395
1.00	0.0261
Average precision over all relevant docs	
non-interpolated	0.2041

Document Level Averages	
	Precision
At 5 docs	0.4680
At 10 docs	0.4360
At 15 docs	0.4027
At 20 docs	0.3610
At 30 docs	0.3120
At 100 docs	0.1928
At 200 docs	0.1317
At 500 docs	0.0726
At 1000 docs	0.0441
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2513

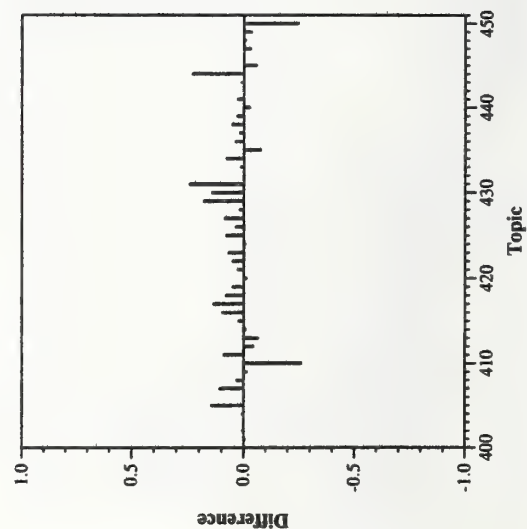
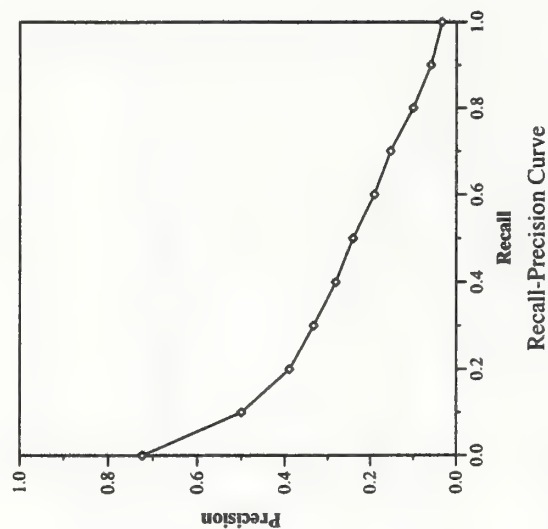


Ad hoc results — Imperial College

Summary Statistics		
Run Number	ic99dafb	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2982	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7245
0.10	0.4996
0.20	0.3894
0.30	0.3333
0.40	0.2815
0.50	0.2408
0.60	0.1911
0.70	0.1532
0.80	0.1005
0.90	0.0594
1.00	0.0339
Average precision over all relevant docs	
non-interpolated	0.2518

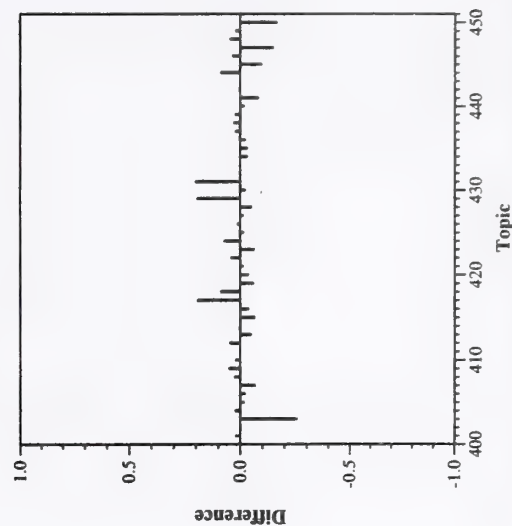
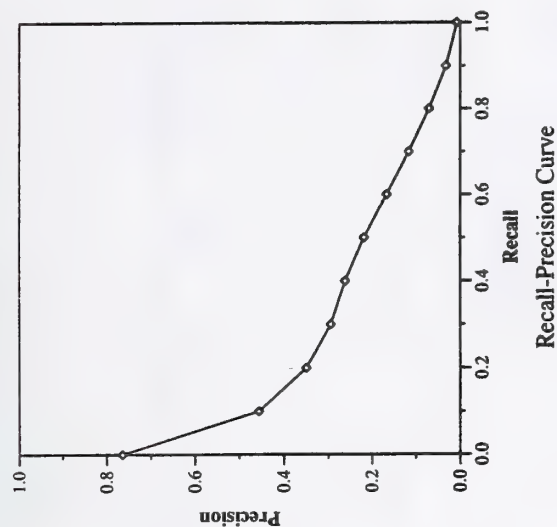
Document Level Averages	
	Precision
At 5 docs	0.4640
At 10 docs	0.4220
At 15 docs	0.3987
At 20 docs	0.3650
At 30 docs	0.3213
At 100 docs	0.2166
At 200 docs	0.1609
At 500 docs	0.0959
At 1000 docs	0.0596
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2906



Summary Statistics	
Run Number	Mer8Adtd1
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	2831

Recall Level Precision Averages	
Recall	Precision
0.00	0.7649
0.10	0.4565
0.20	0.3499
0.30	0.2946
0.40	0.2620
0.50	0.2186
0.60	0.1666
0.70	0.1161
0.80	0.0705
0.90	0.0322
1.00	0.0077
Average precision over all relevant docs	
non-interpolated	0.2231

Document Level Averages	
	Precision
At 5 docs	0.4800
At 10 docs	0.4440
At 15 docs	0.4027
At 20 docs	0.3860
At 30 docs	0.3367
At 100 docs	0.2096
At 200 docs	0.1525
At 500 docs	0.0900
At 1000 docs	0.0566
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2786

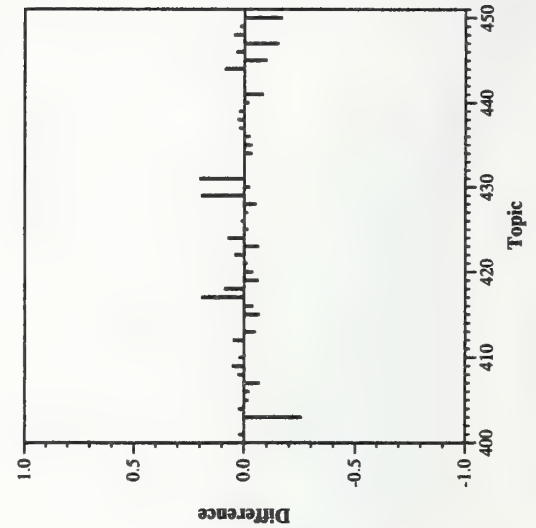
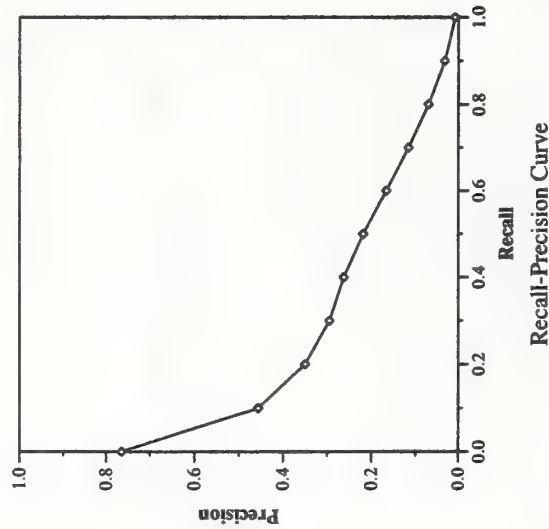


Ad hoc results — IRIIT/SIG

Summary Statistics	
Run Number	Mer8Atdt2
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	2831

Recall Level Precision Averages	
Recall	Precision
0.00	0.7649
0.10	0.4565
0.20	0.3499
0.30	0.2946
0.40	0.2620
0.50	0.2186
0.60	0.1666
0.70	0.1161
0.80	0.0705
0.90	0.0322
1.00	0.0077
Average precision over all relevant docs	
non-interpolated	0.2231

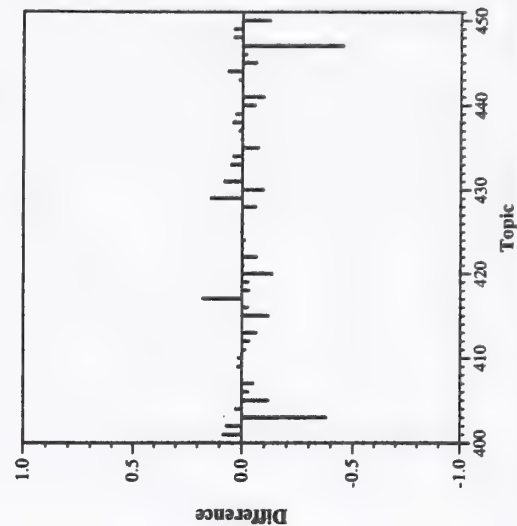
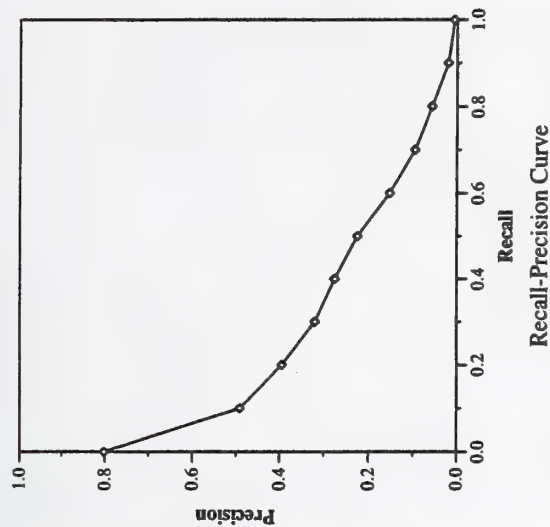
Document Level Averages	
	Precision
At 5 docs	0.4800
At 10 docs	0.4440
At 15 docs	0.4027
At 20 docs	0.3860
At 30 docs	0.3367
At 100 docs	0.2096
At 200 docs	0.1525
At 500 docs	0.0900
At 1000 docs	0.0566
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2786



Summary Statistics		
Run Number	Mer8Adtnd3	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2875	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8021
0.10	0.4908
0.20	0.3943
0.30	0.3198
0.40	0.2757
0.50	0.2252
0.60	0.1532
0.70	0.0957
0.80	0.0572
0.90	0.0202
1.00	0.0063
Average precision over all relevant docs	
non-interpolated	0.2327

Document Level Averages	
	Precision
At 5 docs	0.5480
At 10 docs	0.4620
At 15 docs	0.4147
At 20 docs	0.3890
At 30 docs	0.3540
At 100 docs	0.2288
At 200 docs	0.1620
At 500 docs	0.0952
At 1000 docs	0.0575
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2931

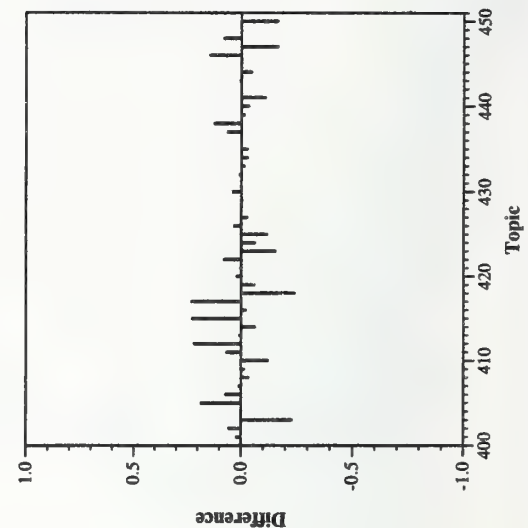
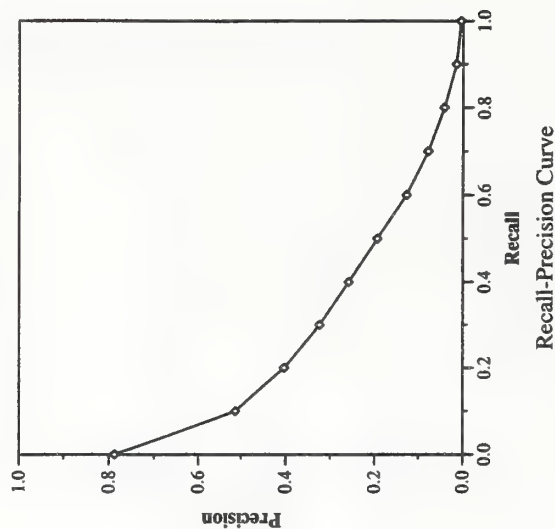


Ad hoc results — IRT/SIG

Summary Statistics		
Run Number	Mer8Adtd4	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2664	

Recall Level Precision Averages		
	Recall	Precision
	0.00	0.7880
	0.10	0.5136
	0.20	0.4032
	0.30	0.3241
	0.40	0.2582
	0.50	0.1934
	0.60	0.1273
	0.70	0.0780
	0.80	0.0417
	0.90	0.0146
	1.00	0.0042
Average precision over all relevant docs		
non-interpolated	0.2247	

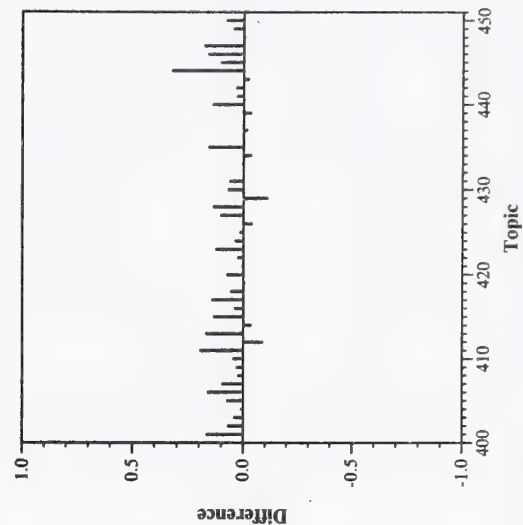
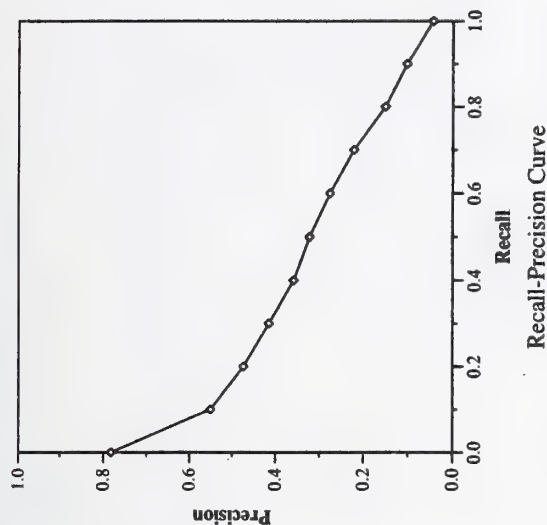
Document Level Averages	
	Precision
At 5 docs	0.5400
At 10 docs	0.4860
At 15 docs	0.4427
At 20 docs	0.4190
At 30 docs	0.3587
At 100 docs	0.2122
At 200 docs	0.1477
At 500 docs	0.0859
At 1000 docs	0.0533
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2733



Summary Statistics		
Run Number	apl8c221	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3332	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7821
0.10	0.5503
0.20	0.4730
0.30	0.4144
0.40	0.3587
0.50	0.3225
0.60	0.2760
0.70	0.2212
0.80	0.1506
0.90	0.1026
1.00	0.0443
Average precision over all relevant docs	
non-interpolated	0.3150

Document Level Averages	
	Precision
At 5 docs	0.5640
At 10 docs	0.5040
At 15 docs	0.4587
At 20 docs	0.4250
At 30 docs	0.3713
At 100 docs	0.2558
At 200 docs	0.1904
At 500 docs	0.1116
At 1000 docs	0.0666
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3374

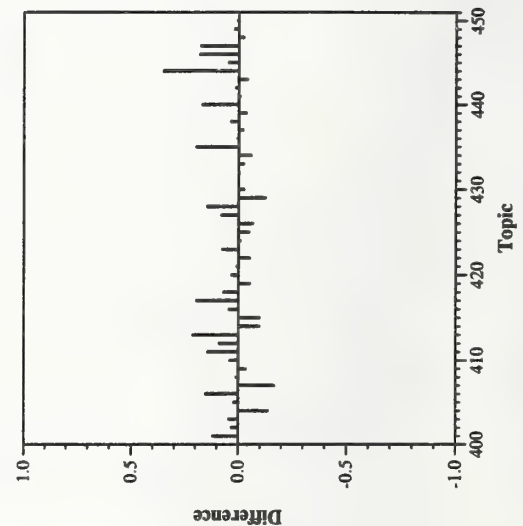
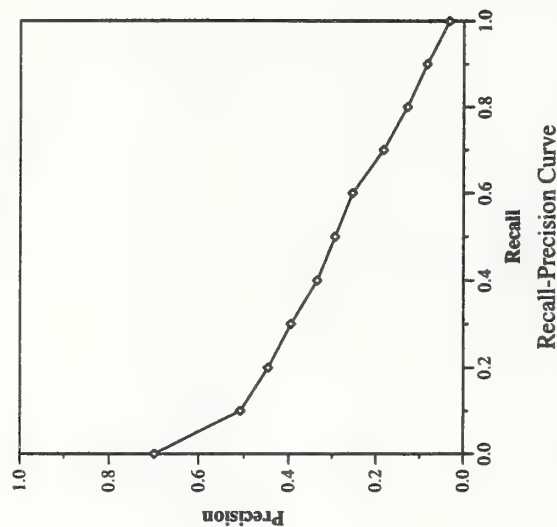


Ad hoc results — Johns Hopkins University

Summary Statistics		
Run Number		apl8n
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3061	

Recall Level Precision Averages		
	Recall	Precision
	0.00	0.6983
	0.10	0.5077
	0.20	0.4463
	0.30	0.3951
	0.40	0.3352
	0.50	0.2938
	0.60	0.2536
	0.70	0.1829
	0.80	0.1288
	0.90	0.0835
	1.00	0.0322
Average precision over all relevant docs		
non-interpolated		0.2885

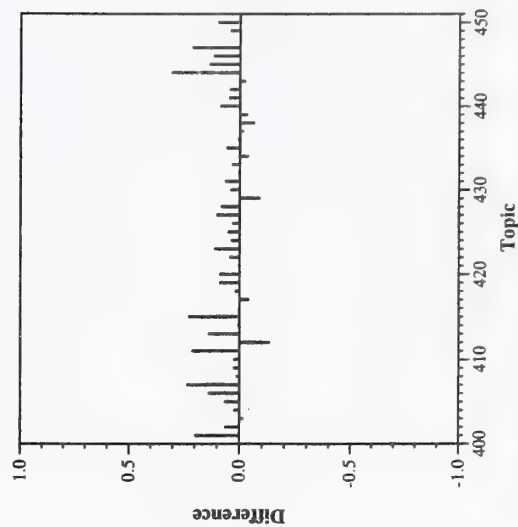
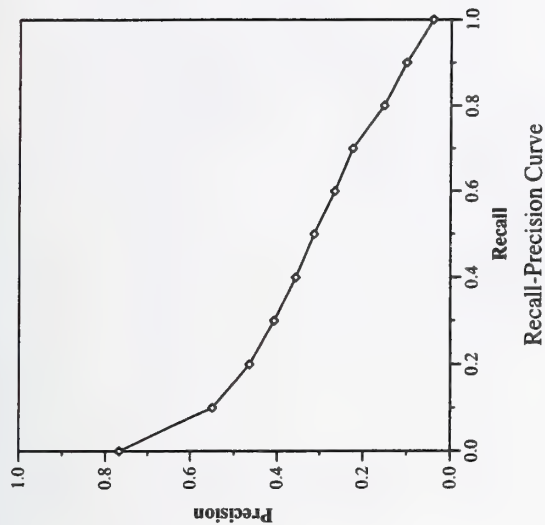
Document Level Averages	
	Precision
At 5 docs	0.4800
At 10 docs	0.4680
At 15 docs	0.4493
At 20 docs	0.4200
At 30 docs	0.3680
At 100 docs	0.2436
At 200 docs	0.1731
At 500 docs	0.1015
At 1000 docs	0.0612
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3162



Summary Statistics	
Run Number	apl8c621
Run Description	Automatic, title + desc + narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3335

Recall Level Precision Averages	
Recall	Precision
0.00	0.7677
0.10	0.5496
0.20	0.4641
0.30	0.4070
0.40	0.3570
0.50	0.3139
0.60	0.2663
0.70	0.2244
0.80	0.1522
0.90	0.1017
1.00	0.0410
Average precision over all relevant docs	
non-interpolated	0.3126

Document Level Averages	
	Precision
At 5 docs	0.5600
At 10 docs	0.5040
At 15 docs	0.4627
At 20 docs	0.4170
At 30 docs	0.3707
At 100 docs	0.2558
At 200 docs	0.1900
At 500 docs	0.1099
At 1000 docs	0.0667
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3380

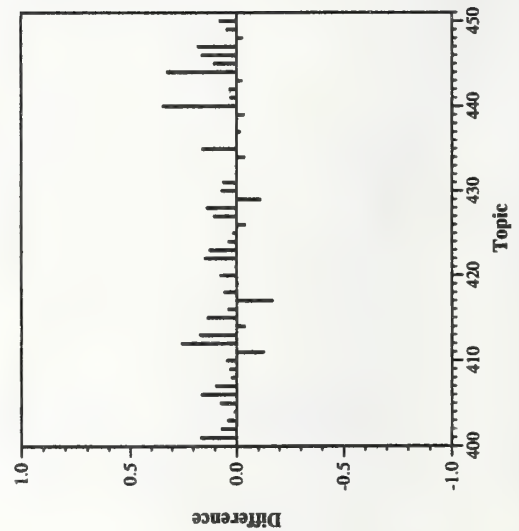
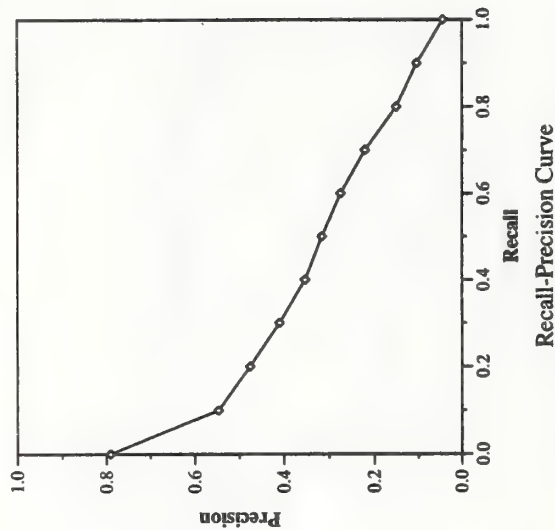


Ad hoc results — Johns Hopkins University

Summary Statistics		
Run Number	apl8p	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3295	

Recall Level Precision Averages		
Recall	Precision	
0.00	0.7913	
0.10	0.5473	
0.20	0.4761	
0.30	0.4100	
0.40	0.3534	
0.50	0.3162	
0.60	0.2746	
0.70	0.2212	
0.80	0.1511	
0.90	0.1042	
1.00	0.0443	
Average precision over all relevant docs		
non-interpolated	0.3154	

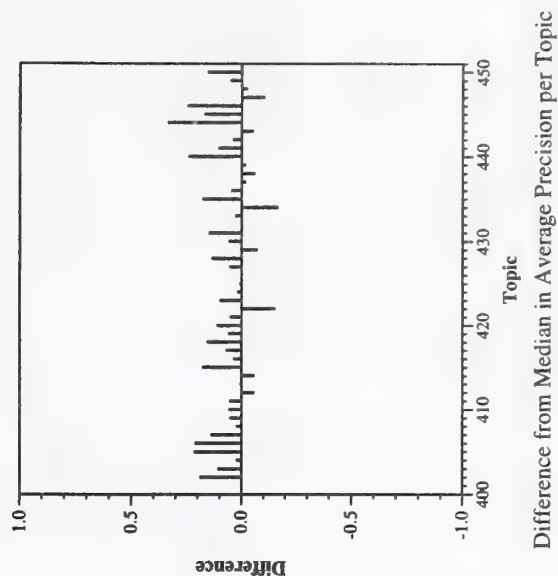
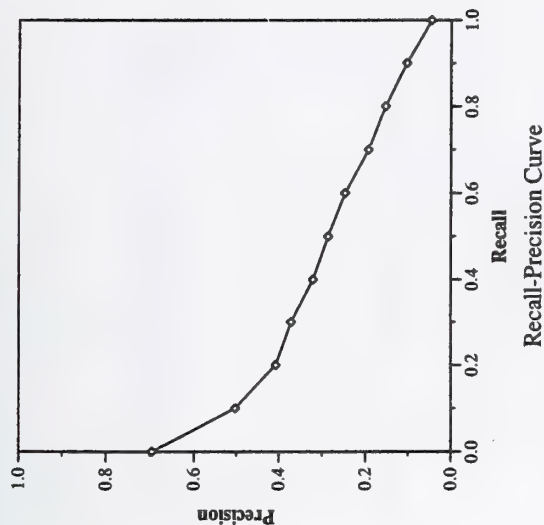
Document Level Averages	
	Precision
At 5 docs	0.5760
At 10 docs	0.5160
At 15 docs	0.4693
At 20 docs	0.4310
At 30 docs	0.3747
At 100 docs	0.2568
At 200 docs	0.1912
At 500 docs	0.1104
At 1000 docs	0.0659
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3290



Summary Statistics		
Run Number	ap18ctd	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3117	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6963
0.10	0.5030
0.20	0.4074
0.30	0.3714
0.40	0.3215
0.50	0.2866
0.60	0.2477
0.70	0.1933
0.80	0.1530
0.90	0.1031
1.00	0.0449
Average precision over all relevant docs	
non-interpolated	0.2860

Document Level Averages	
	Precision
At 5 docs	0.4960
At 10 docs	0.4500
At 15 docs	0.4240
At 20 docs	0.3910
At 30 docs	0.3447
At 100 docs	0.2324
At 200 docs	0.1723
At 500 docs	0.1026
At 1000 docs	0.0623
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3099

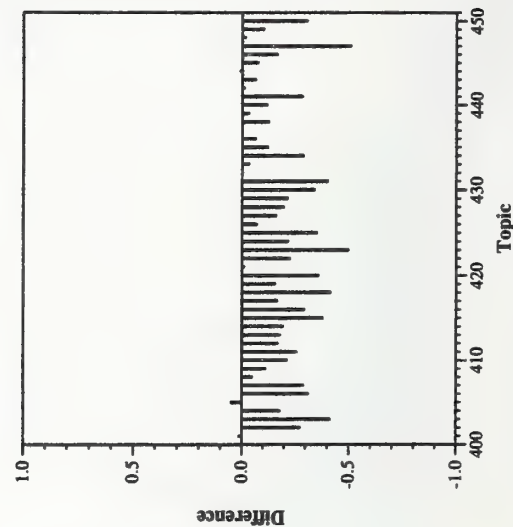
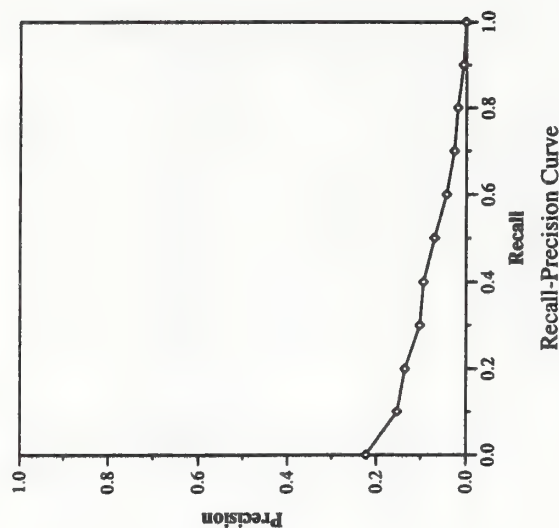


Ad hoc results — Kasetsart University

Summary Statistics		
Run Number	kuadhoc	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	1202	

Recall Level Precision Averages	
Recall	Precision
0.00	0.2229
0.10	0.1528
0.20	0.1354
0.30	0.1022
0.40	0.0939
0.50	0.0698
0.60	0.0437
0.70	0.0267
0.80	0.0192
0.90	0.0063
1.00	0.0009
Average precision over all relevant docs	
non-interpolated	0.0698

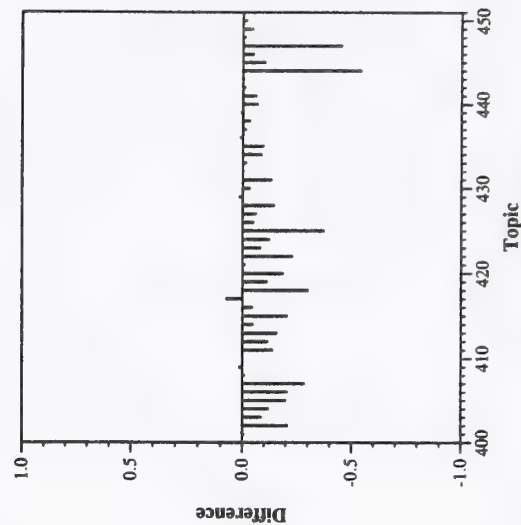
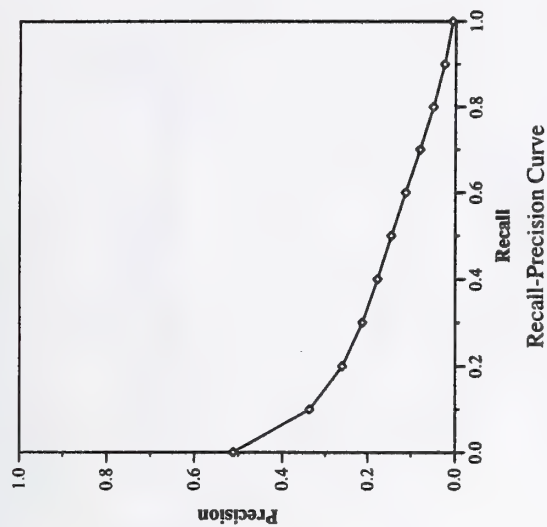
Document Level Averages	
	Precision
At 5 docs	0.1320
At 10 docs	0.1360
At 15 docs	0.1200
At 20 docs	0.1110
At 30 docs	0.1013
At 100 docs	0.0658
At 200 docs	0.0489
At 500 docs	0.0328
At 1000 docs	0.0240
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0970



Summary Statistics		
Run Number	kdd8ps16	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2351	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5114
0.10	0.3368
0.20	0.2604
0.30	0.2129
0.40	0.1773
0.50	0.1450
0.60	0.1126
0.70	0.0801
0.80	0.0502
0.90	0.0251
1.00	0.0072
Average precision over all relevant docs	
non-interpolated	0.1542

Document Level Averages	
	Precision
At 5 docs	0.3120
At 10 docs	0.2800
At 15 docs	0.2533
At 20 docs	0.2530
At 30 docs	0.2240
At 100 docs	0.1536
At 200 docs	0.1143
At 500 docs	0.0710
At 1000 docs	0.0470
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2089

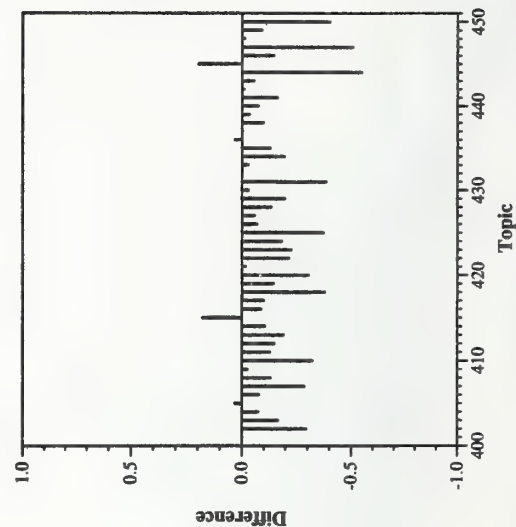
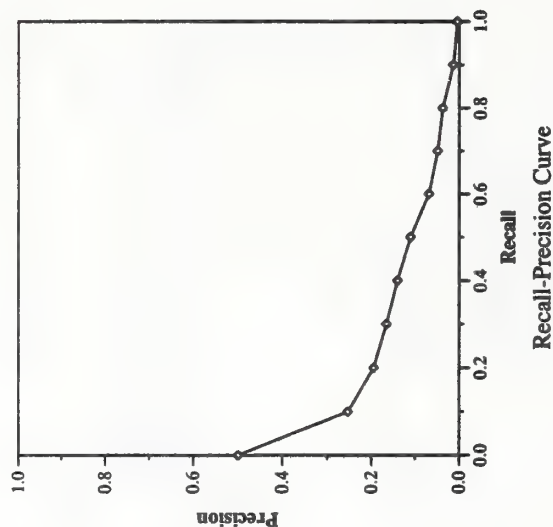


Ad hoc results — KDD R&D Laboratories

Summary Statistics		
Run Number	kdd8qe01	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	1665	

Recall Level Precision Averages	
Recall	Precision
0.00	0.4981
0.10	0.2538
0.20	0.1946
0.30	0.1653
0.40	0.1401
0.50	0.1099
0.60	0.0687
0.70	0.0489
0.80	0.0374
0.90	0.0135
1.00	0.0040
Average precision over all relevant docs	
non-interpolated	0.1187

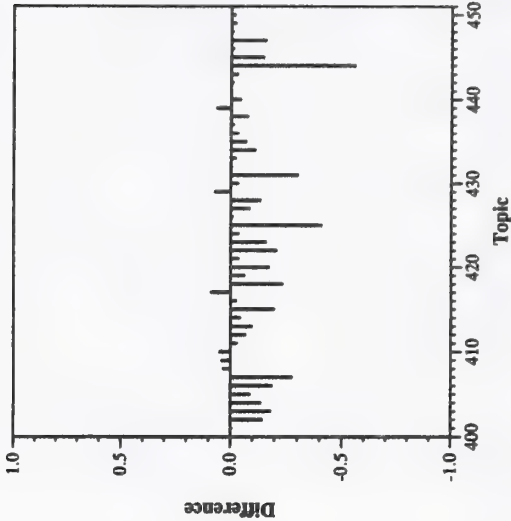
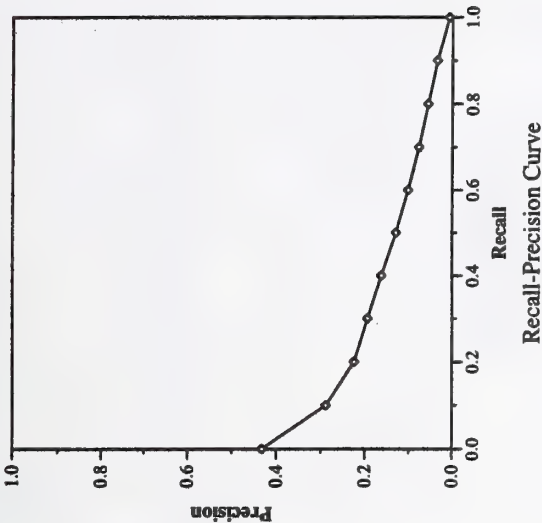
Document Level Averages	
	Precision
At 5 docs	0.2880
At 10 docs	0.2820
At 15 docs	0.2413
At 20 docs	0.2120
At 30 docs	0.1787
At 100 docs	0.1130
At 200 docs	0.0774
At 500 docs	0.0486
At 1000 docs	0.0333
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1580



Summary Statistics	
Run Number	kdd8sh16
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	2178

Recall Level Precision Averages	
Recall	Precision
0.00	0.4328
0.10	0.2885
0.20	0.2246
0.30	0.1936
0.40	0.1627
0.50	0.1296
0.60	0.1022
0.70	0.0774
0.80	0.0566
0.90	0.0350
1.00	0.0073
Average precision over all relevant docs	
non-interpolated	0.1397

Document Level Averages	
	Precision
At 5 docs	0.2520
At 10 docs	0.2520
At 15 docs	0.2413
At 20 docs	0.2290
At 30 docs	0.1953
At 100 docs	0.1328
At 200 docs	0.0986
At 500 docs	0.0638
At 1000 docs	0.0436
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1904

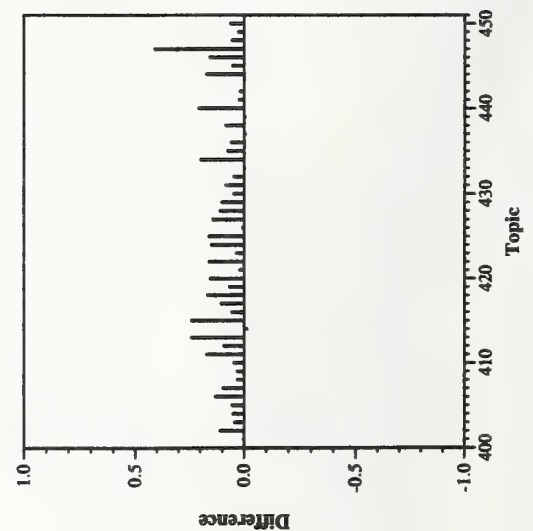
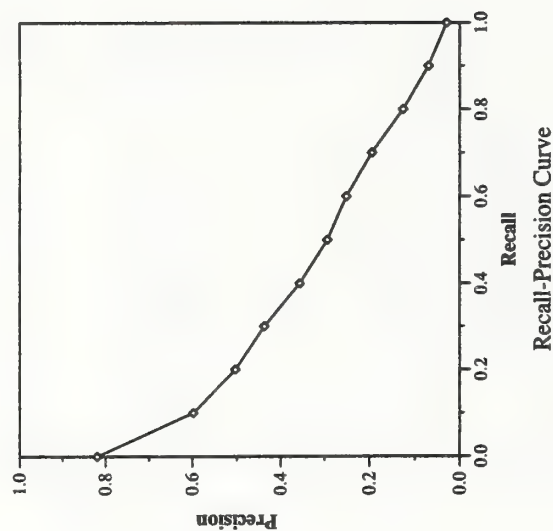


Difference from Median in Average Precision per Topic

Summary Statistics		
Run Number	ok8amxc	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3212	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8190
0.10	0.5975
0.20	0.5032
0.30	0.4372
0.40	0.3561
0.50	0.2936
0.60	0.2511
0.70	0.1941
0.80	0.1257
0.90	0.0696
1.00	0.0296
Average precision over all relevant docs	
non-interpolated	0.3169

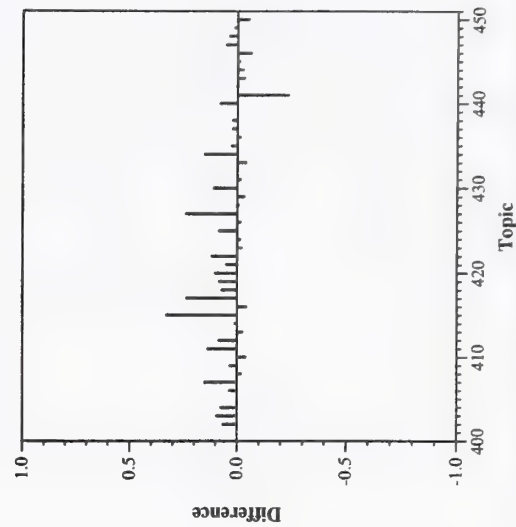
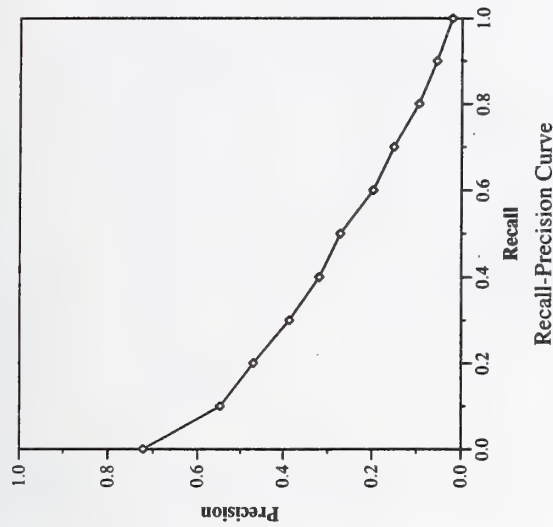
Document Level Averages	
	Precision
At 5 docs	0.5800
At 10 docs	0.5500
At 15 docs	0.4987
At 20 docs	0.4650
At 30 docs	0.4253
At 100 docs	0.2680
At 200 docs	0.1921
At 500 docs	0.1085
At 1000 docs	0.0642
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3470



Summary Statistics	
Run Number	ok8asxc
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3013

Recall Level Precision Averages	
Recall	Precision
0.00	0.7220
0.10	0.5482
0.20	0.4698
0.30	0.3860
0.40	0.3187
0.50	0.2717
0.60	0.1979
0.70	0.1516
0.80	0.0957
0.90	0.0556
1.00	0.0215
Average precision over all relevant docs	
non-interpolated	0.2787

Document Level Averages	
	Precision
At 5 docs	0.5520
At 10 docs	0.4880
At 15 docs	0.4480
At 20 docs	0.4160
At 30 docs	0.3800
At 100 docs	0.2450
At 200 docs	0.1782
At 500 docs	0.1012
At 1000 docs	0.0603
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3088

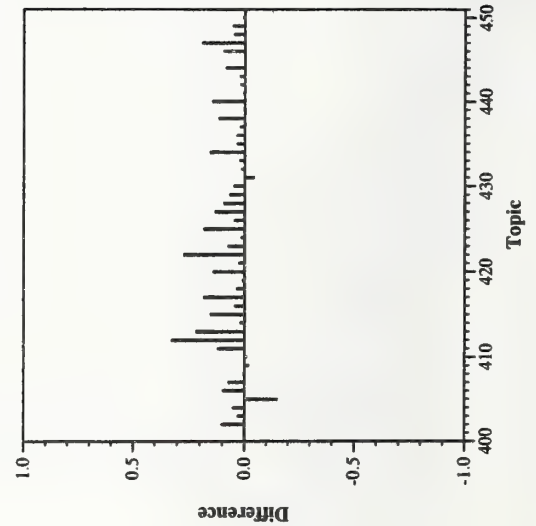
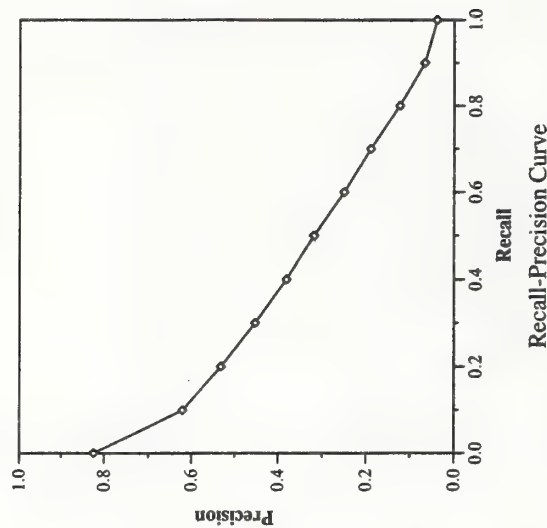


Ad hoc results — Microsoft Research Ltd

Summary Statistics		
Run Number	ok8alx	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3282	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8239
0.10	0.6206
0.20	0.5321
0.30	0.4529
0.40	0.3806
0.50	0.3181
0.60	0.2501
0.70	0.1888
0.80	0.1228
0.90	0.0662
1.00	0.0389
Average precision over all relevant docs	
non-interpolated	0.3240

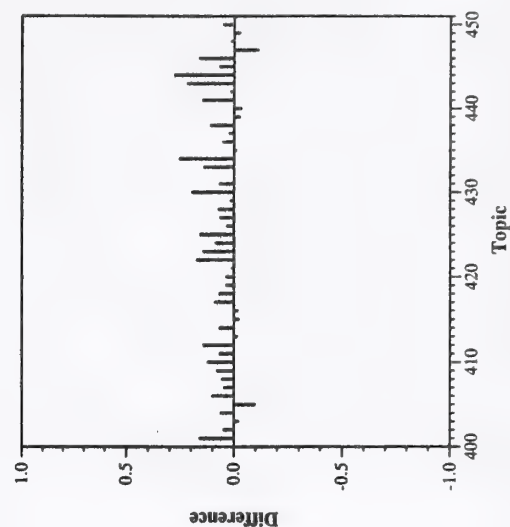
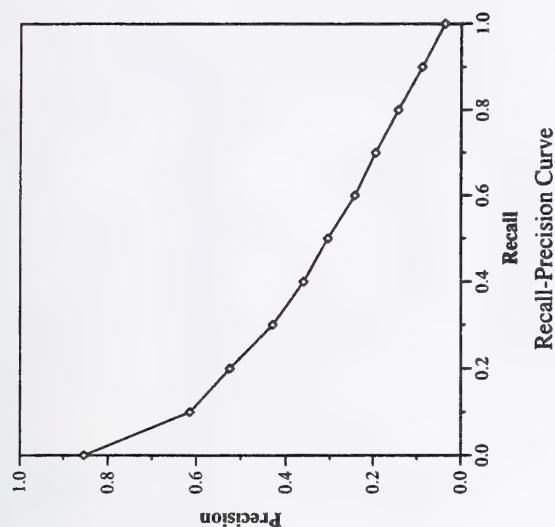
Document Level Averages	
	Precision
At 5 docs	0.6360
At 10 docs	0.5700
At 15 docs	0.5173
At 20 docs	0.4830
At 30 docs	0.4433
At 100 docs	0.2724
At 200 docs	0.1957
At 500 docs	0.1106
At 1000 docs	0.0656
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3542



Summary Statistics		
Run Number	MITSLStdN	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3318	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8537
0.10	0.6144
0.20	0.5248
0.30	0.4293
0.40	0.3604
0.50	0.3052
0.60	0.2428
0.70	0.1948
0.80	0.1427
0.90	0.0884
1.00	0.0366
Average precision over all relevant docs	
non-interpolated	0.3227

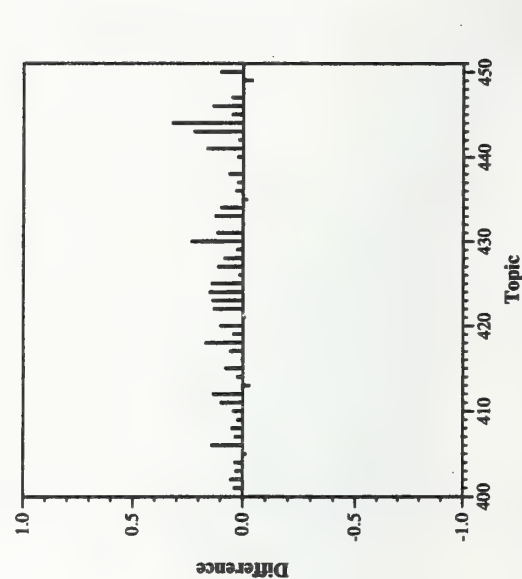
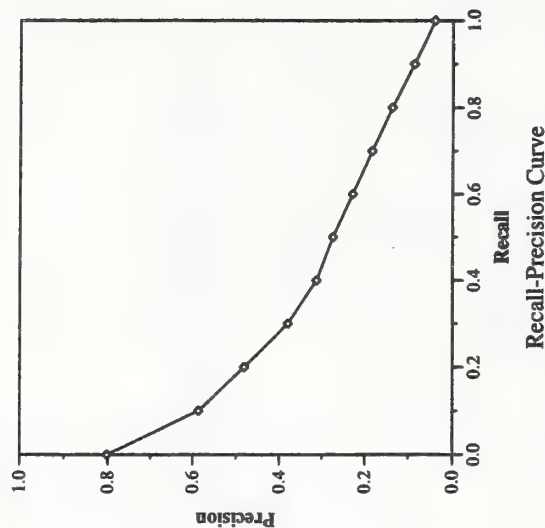
Document Level Averages	
	Precision
At 5 docs	0.6120
At 10 docs	0.5480
At 15 docs	0.5213
At 20 docs	0.4890
At 30 docs	0.4327
At 100 docs	0.2772
At 200 docs	0.1950
At 500 docs	0.1119
At 1000 docs	0.0664
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3587



Ad hoc results — MIT Laboratory for Computer Science

Summary Statistics		
Run Number	MITSLStd	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3114	

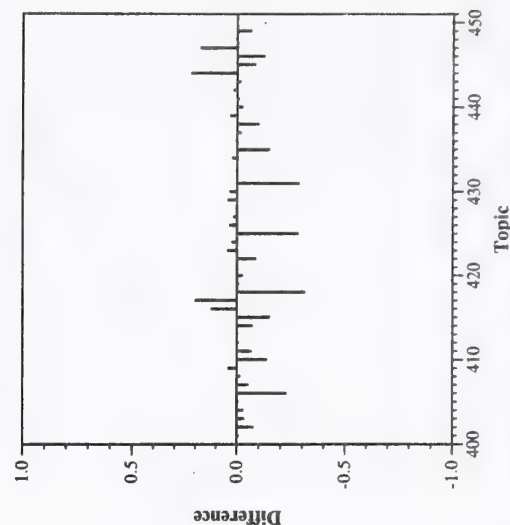
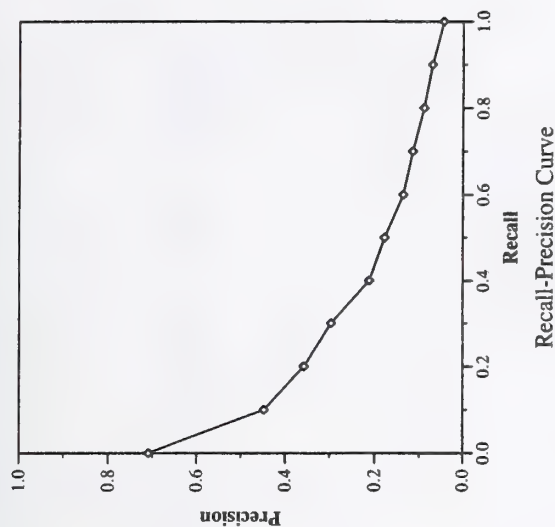
Recall Level Precision Averages		Document Level Averages	
Recall	Precision		Precision
0.00	0.8026	At 5 docs	0.5720
0.10	0.5867	At 10 docs	0.5080
0.20	0.4817	At 15 docs	0.4720
0.30	0.3805	At 20 docs	0.4480
0.40	0.3134	At 30 docs	0.4073
0.50	0.2747	At 100 docs	0.2558
0.60	0.2292	At 200 docs	0.1801
0.70	0.1836	At 500 docs	0.1029
0.80	0.1373	At 1000 docs	0.0623
0.90	0.0868	R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
1.00	0.0404		
Average precision over all relevant docs		Exact	0.3271



Summary Statistics		
Run Number	uwmt8a0	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	48768	
Relevant:	4728	
Rel-ret:	2441	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7096
0.10	0.4480
0.20	0.3579
0.30	0.2969
0.40	0.2121
0.50	0.1783
0.60	0.1371
0.70	0.1157
0.80	0.0905
0.90	0.0713
1.00	0.0454
Average precision over all relevant docs	
non-interpolated	0.2143

Document Level Averages	
	Precision
At 5 docs	0.4640
At 10 docs	0.4080
At 15 docs	0.3640
At 20 docs	0.3410
At 30 docs	0.3120
At 100 docs	0.1972
At 200 docs	0.1384
At 500 docs	0.0757
At 1000 docs	0.0488
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2635

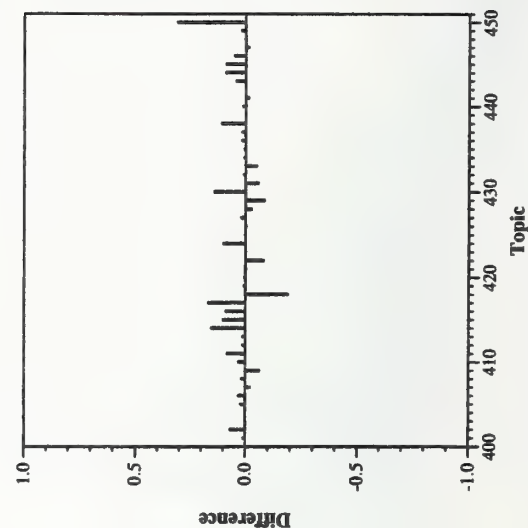
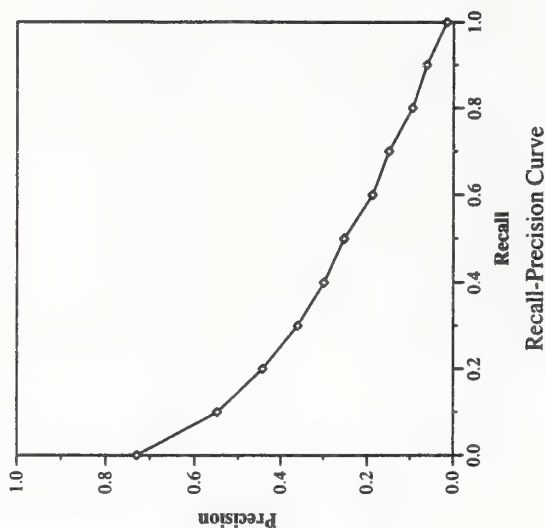


Ad hoc results — MultiText Project

Summary Statistics	
Run Number	uwmt8a1
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	2979

Recall Level Precision Averages	
Recall	Precision
0.00	0.7299
0.10	0.5486
0.20	0.4430
0.30	0.3599
0.40	0.2989
0.50	0.2526
0.60	0.1878
0.70	0.1502
0.80	0.0946
0.90	0.0619
1.00	0.0153
Average precision over all relevant docs	
non-interpolated	0.2673

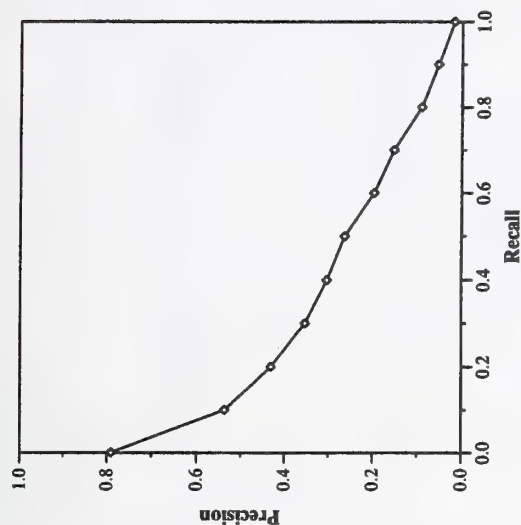
Document Level Averages	
	Precision
At 5 docs	0.5560
At 10 docs	0.4820
At 15 docs	0.4613
At 20 docs	0.4250
At 30 docs	0.3827
At 100 docs	0.2412
At 200 docs	0.1696
At 500 docs	0.0983
At 1000 docs	0.0596
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3002



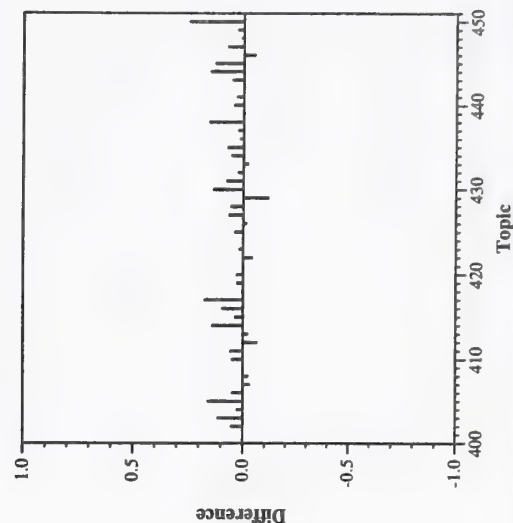
Summary Statistics		
Run Number	uwmt8a2	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2902	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7905
0.10	0.5367
0.20	0.4311
0.30	0.3534
0.40	0.3034
0.50	0.2637
0.60	0.1973
0.70	0.1525
0.80	0.0901
0.90	0.0528
1.00	0.0172
Average precision over all relevant docs	
non-interpolated	0.2671

Document Level Averages	
	Precision
At 5 docs	0.5280
At 10 docs	0.4840
At 15 docs	0.4427
At 20 docs	0.4020
At 30 docs	0.3567
At 100 docs	0.2314
At 200 docs	0.1659
At 500 docs	0.0947
At 1000 docs	0.0580
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3046



Recall-Precision Curve



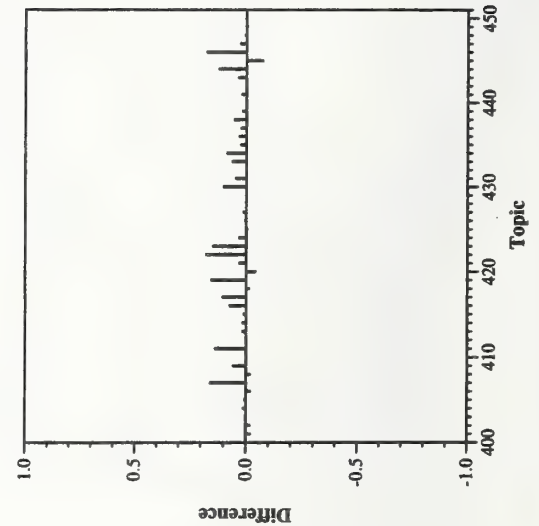
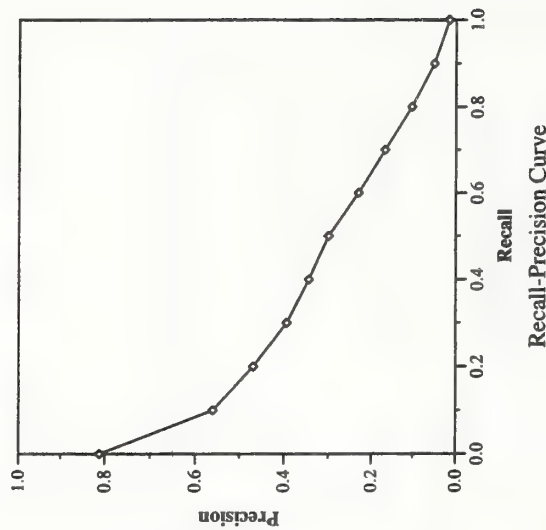
Difference from Median in Average Precision per Topic

Ad hoc results — NTT DATA Corporation

Summary Statistics		
Run Number	nttd8ale	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3120	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8119
0.10	0.5605
0.20	0.4696
0.30	0.3937
0.40	0.3443
0.50	0.2994
0.60	0.2304
0.70	0.1676
0.80	0.1032
0.90	0.0503
1.00	0.0157
Average precision over all relevant docs	
non-interpolated	0.2921

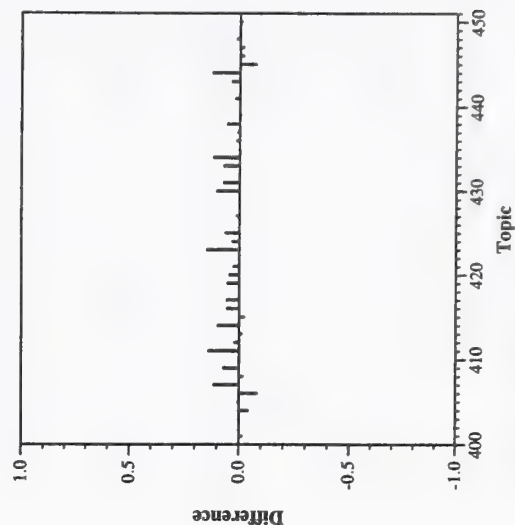
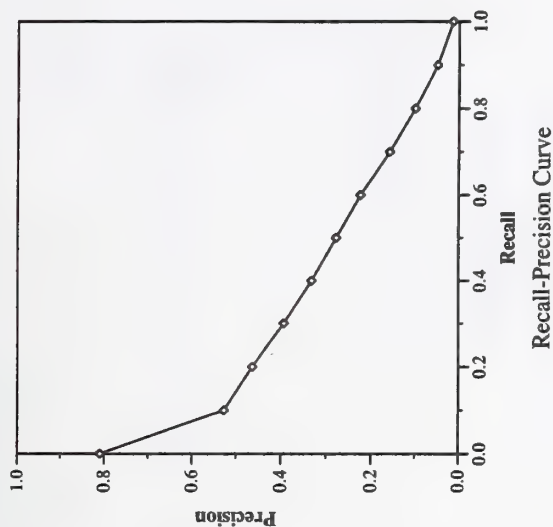
Document Level Averages	
	Precision
At 5 docs	0.5440
At 10 docs	0.4940
At 15 docs	0.4720
At 20 docs	0.4370
At 30 docs	0.3847
At 100 docs	0.2506
At 200 docs	0.1808
At 500 docs	0.1038
At 1000 docs	0.0624
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3324



Summary Statistics		
Run Number	nttd8alx	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2986	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8097
0.10	0.5273
0.20	0.4635
0.30	0.3935
0.40	0.3323
0.50	0.2781
0.60	0.2240
0.70	0.1580
0.80	0.0997
0.90	0.0485
1.00	0.0136
Average precision over all relevant docs	
non-interpolated	0.2817

Document Level Averages	
	Precision
At 5 docs	0.5480
At 10 docs	0.4760
At 15 docs	0.4533
At 20 docs	0.4230
At 30 docs	0.3840
At 100 docs	0.2460
At 200 docs	0.1759
At 500 docs	0.1007
At 1000 docs	0.0597
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3294

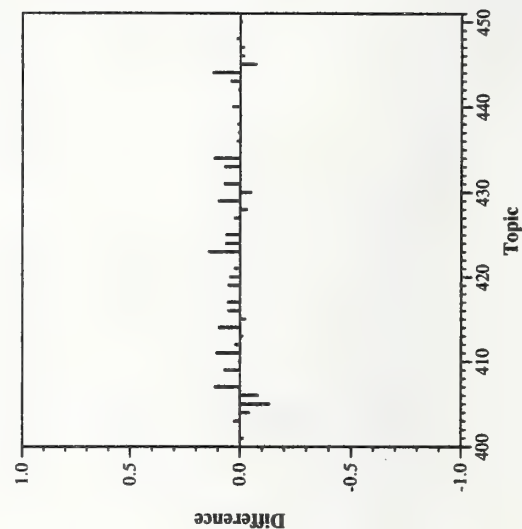
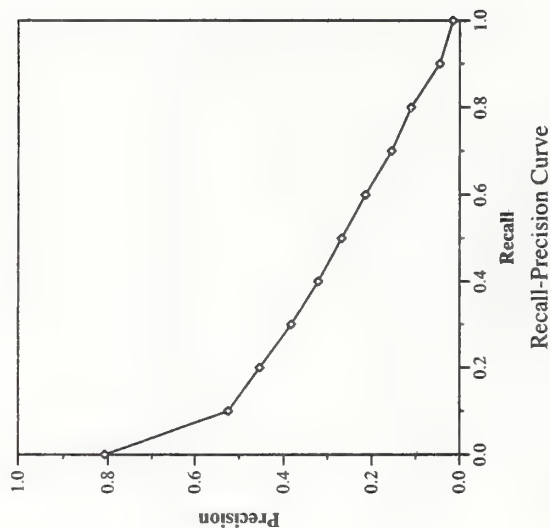


Ad hoc results — NTT DATA Corporation

Summary Statistics		
Run Number	nttd8al	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2973	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8072
0.10	0.5247
0.20	0.4545
0.30	0.3826
0.40	0.3214
0.50	0.2682
0.60	0.2149
0.70	0.1552
0.80	0.1108
0.90	0.0459
1.00	0.0155
Average precision over all relevant docs	
non-interpolated	0.2781

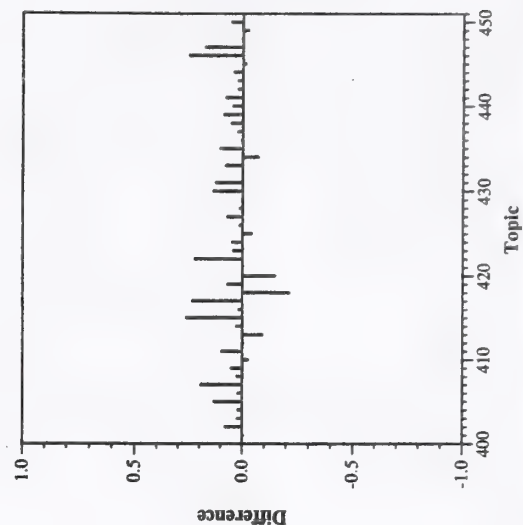
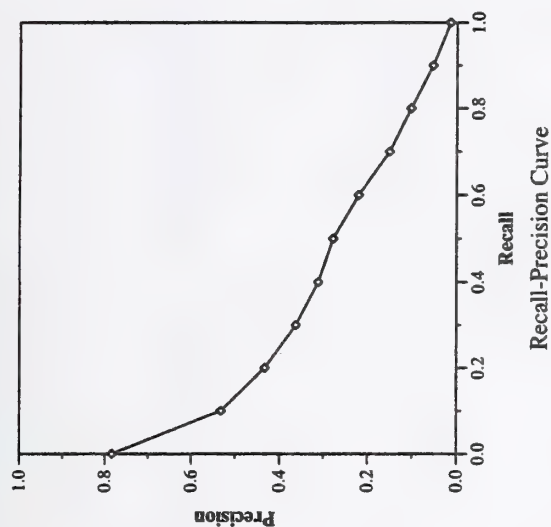
Document Level Averages	
	Precision
At 5 docs	0.5480
At 10 docs	0.4880
At 15 docs	0.4600
At 20 docs	0.4220
At 30 docs	0.3800
At 100 docs	0.2450
At 200 docs	0.1730
At 500 docs	0.1000
At 1000 docs	0.0595
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3247



Summary Statistics		
Run Number	nttd8ame	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3028	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7835
0.10	0.5319
0.20	0.4341
0.30	0.3629
0.40	0.3116
0.50	0.2779
0.60	0.2202
0.70	0.1510
0.80	0.1025
0.90	0.0539
1.00	0.0151
Average precision over all relevant docs	
non-interpolated	0.2721

Document Level Averages	
	Precision
At 5 docs	0.4920
At 10 docs	0.4900
At 15 docs	0.4493
At 20 docs	0.4070
At 30 docs	0.3747
At 100 docs	0.2384
At 200 docs	0.1706
At 500 docs	0.1004
At 1000 docs	0.0606
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3156



Ad hoc results — NTT DATA Corporation

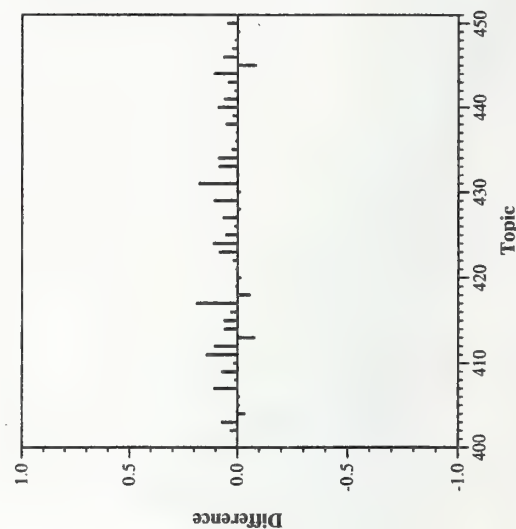
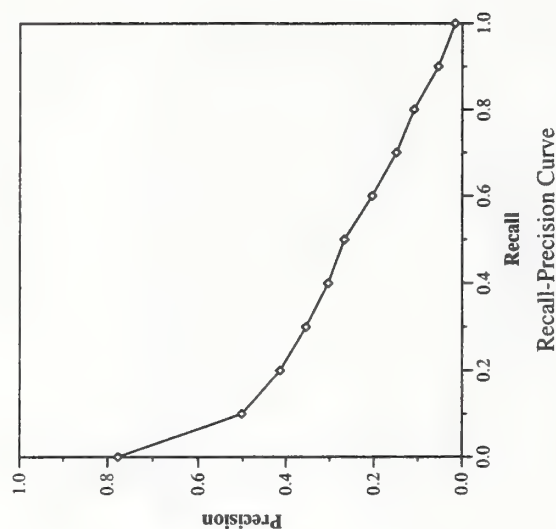
Summary Statistics		
Run Number	nttd8am	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2937	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7765
0.10	0.5028
0.20	0.4142
0.30	0.3539
0.40	0.3038
0.50	0.2672
0.60	0.2044
0.70	0.1491
0.80	0.1091
0.90	0.0536
1.00	0.0161

Average precision over all relevant docs	
non-interpolated	0.2649

Document Level Averages	
	Precision
At 5 docs	0.5080
At 10 docs	0.4600
At 15 docs	0.4307
At 20 docs	0.4030
At 30 docs	0.3667
At 100 docs	0.2422
At 200 docs	0.1706
At 500 docs	0.0984
At 1000 docs	0.0587

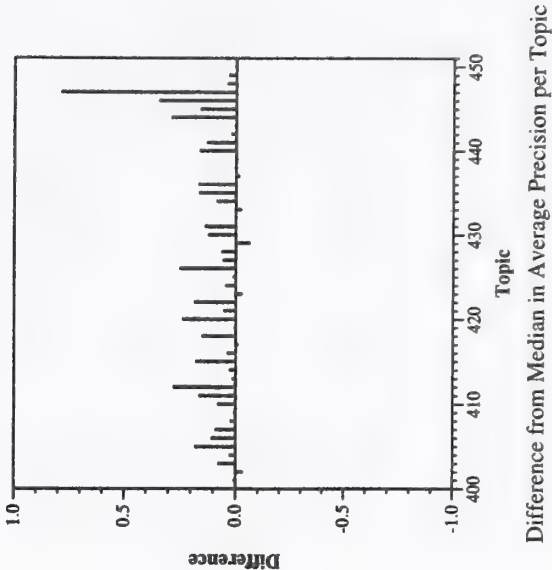
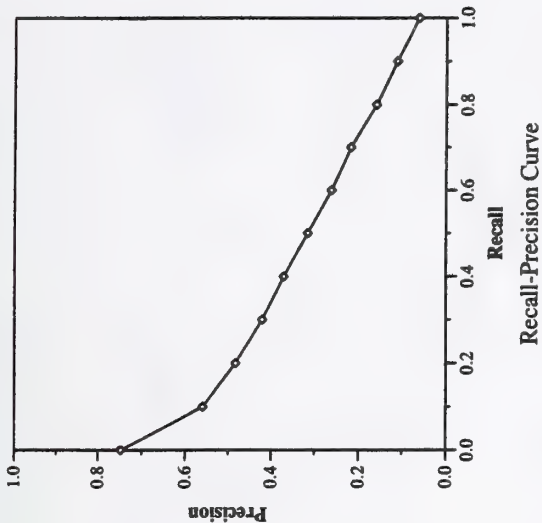
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3157



Summary Statistics		
Run Number	pir9Attd	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3342	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7475
0.10	0.5586
0.20	0.4829
0.30	0.4207
0.40	0.3710
0.50	0.3156
0.60	0.2618
0.70	0.2182
0.80	0.1596
0.90	0.1120
1.00	0.0629
Average precision over all relevant docs	
non-interpolated	0.3207

Document Level Averages	
	Precision
At 5 docs	0.5480
At 10 docs	0.5080
At 15 docs	0.4773
At 20 docs	0.4450
At 30 docs	0.4033
At 100 docs	0.2732
At 200 docs	0.1951
At 500 docs	0.1117
At 1000 docs	0.0668
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3441

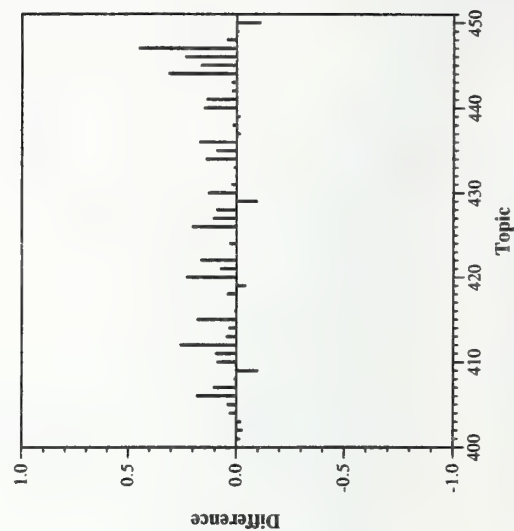
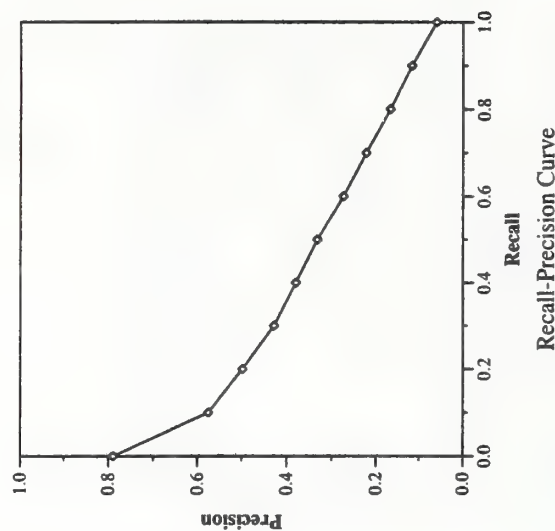


Ad hoc results — Queens College, CUNY

Summary Statistics		
Run Number	pir9Aatd	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3382	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7894
0.10	0.5753
0.20	0.4985
0.30	0.4279
0.40	0.3800
0.50	0.3324
0.60	0.2742
0.70	0.2227
0.80	0.1668
0.90	0.1164
1.00	0.0600
Average precision over all relevant docs	
non-interpolated	0.3303

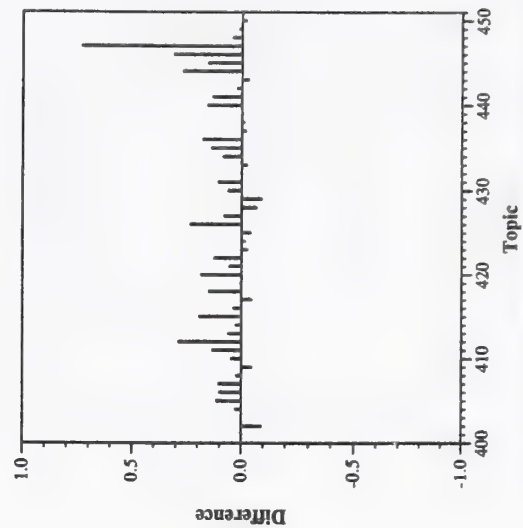
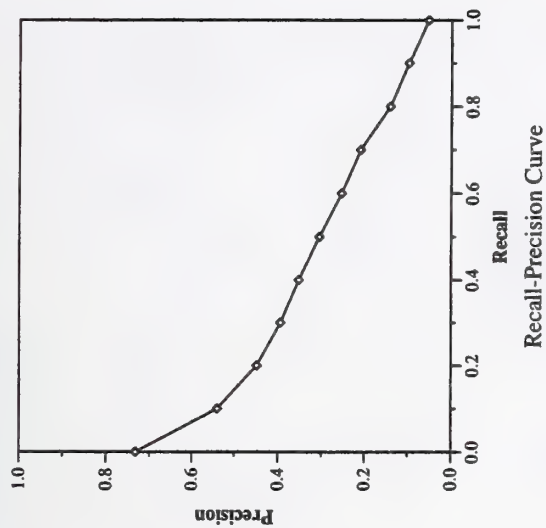
Document Level Averages	
	Precision
At 5 docs	0.5960
At 10 docs	0.5120
At 15 docs	0.4947
At 20 docs	0.4600
At 30 docs	0.4120
At 100 docs	0.2752
At 200 docs	0.1975
At 500 docs	0.1125
At 1000 docs	0.0676
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3512



Summary Statistics		
Run Number	pir9Atd0	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3272	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7323
0.10	0.5394
0.20	0.4483
0.30	0.3943
0.40	0.3524
0.50	0.3051
0.60	0.2537
0.70	0.2097
0.80	0.1405
0.90	0.0978
1.00	0.0524
Average precision over all relevant docs	
non-interpolated	0.3022

Document Level Averages	
	Precision
At 5 docs	0.5400
At 10 docs	0.4920
At 15 docs	0.4547
At 20 docs	0.4290
At 30 docs	0.3807
At 100 docs	0.2574
At 200 docs	0.1876
At 500 docs	0.1084
At 1000 docs	0.0654
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3301

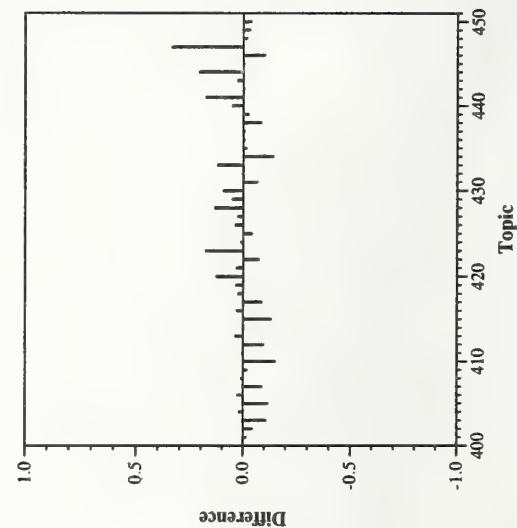
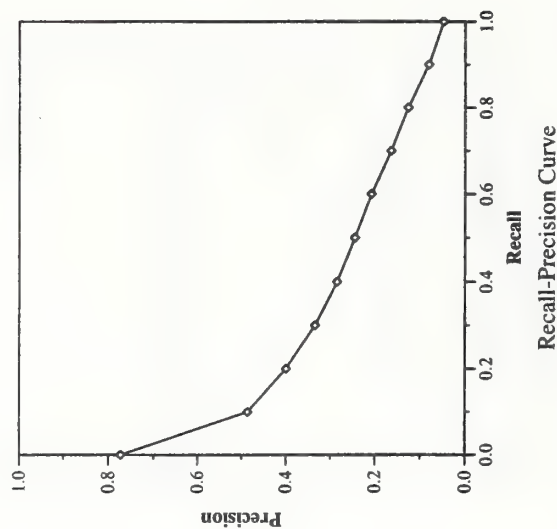


Ad hoc results — Queens College, CUNY

Summary Statistics		
Run Number	pir9Aa1	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2751	

Recall Level Precision Averages		
Recall		Precision
0.00		0.7732
0.10		0.4867
0.20		0.4000
0.30		0.3345
0.40		0.2849
0.50		0.2447
0.60		0.2076
0.70		0.1635
0.80		0.1259
0.90		0.0805
1.00		0.0483
Average precision over all relevant docs		
non-interpolated		0.2624

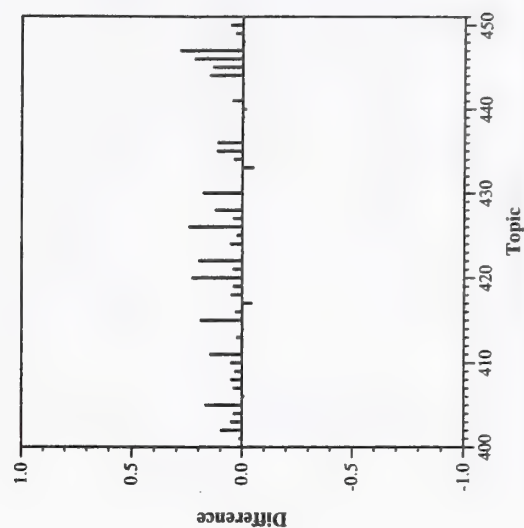
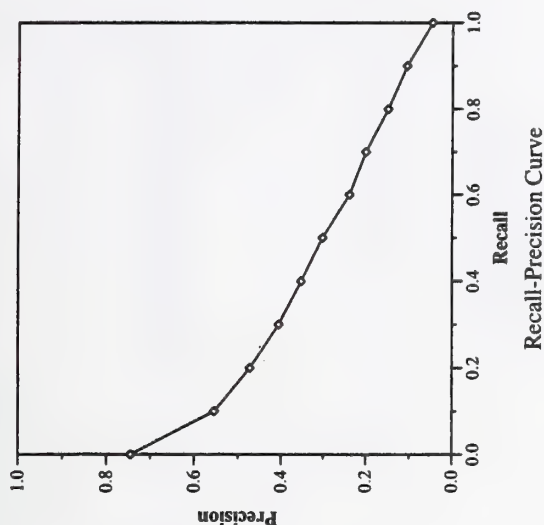
Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.4500
At 15 docs	0.4147
At 20 docs	0.3860
At 30 docs	0.3327
At 100 docs	0.2128
At 200 docs	0.1578
At 500 docs	0.0926
At 1000 docs	0.0550
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3065



Summary Statistics	
Run Number	pir9At0
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3299

Recall Level Precision Averages	
Recall	Precision
0.00	0.7455
0.10	0.5543
0.20	0.4734
0.30	0.4062
0.40	0.3533
0.50	0.3023
0.60	0.2386
0.70	0.2006
0.80	0.1496
0.90	0.1061
1.00	0.0471
Average precision over all relevant docs	
non-interpolated	0.3063

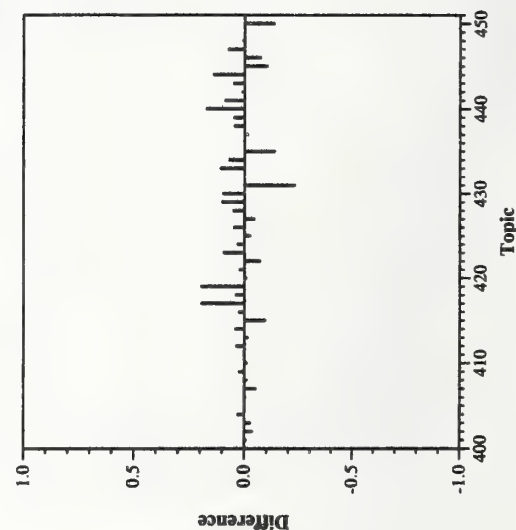
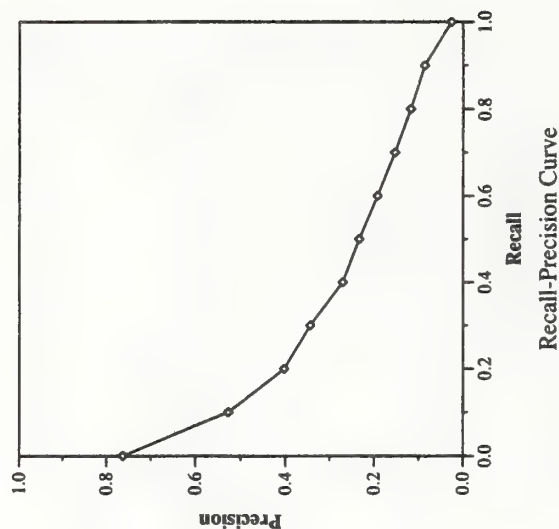
Document Level Averages	
	Precision
At 5 docs	0.5160
At 10 docs	0.4800
At 15 docs	0.4547
At 20 docs	0.4410
At 30 docs	0.4027
At 100 docs	0.2660
At 200 docs	0.1903
At 500 docs	0.1096
At 1000 docs	0.0660
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3326



Summary Statistics		
Run Number	ric8tpn	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	49085	
Relevant:	4728	
Rel-ret:	2707	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7632
0.10	0.5276
0.20	0.4033
0.30	0.3442
0.40	0.2701
0.50	0.2330
0.60	0.1924
0.70	0.1534
0.80	0.1179
0.90	0.0867
1.00	0.0263
Average precision over all relevant docs	
non-interpolated	0.2572

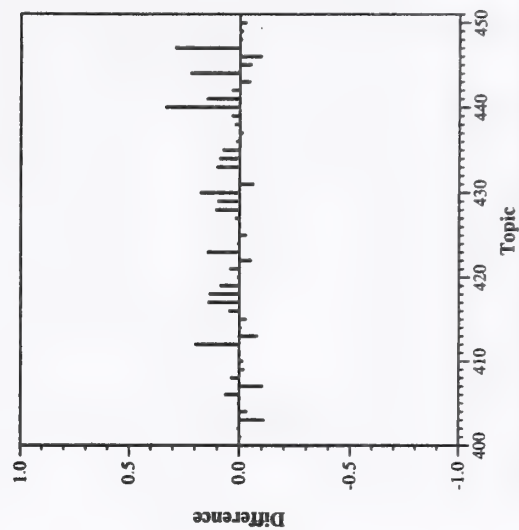
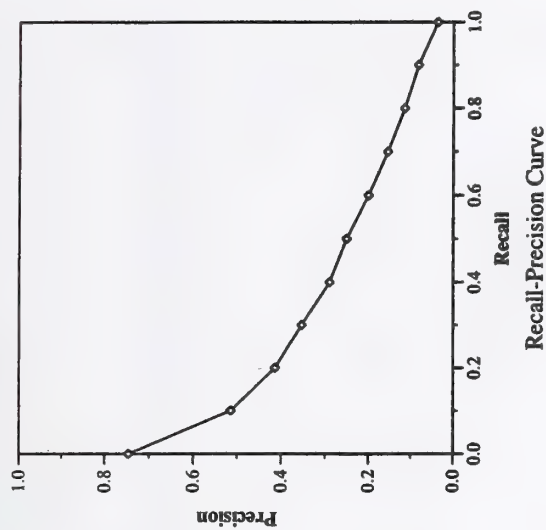
Document Level Averages	
	Precision
At 5 docs	0.4960
At 10 docs	0.4380
At 15 docs	0.4227
At 20 docs	0.3940
At 30 docs	0.3493
At 100 docs	0.2278
At 200 docs	0.1609
At 500 docs	0.0902
At 1000 docs	0.0541
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3081



Summary Statistics		
Run Number	ric8dpm	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2869	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7466
0.10	0.5142
0.20	0.4136
0.30	0.3533
0.40	0.2894
0.50	0.2505
0.60	0.2001
0.70	0.1545
0.80	0.1146
0.90	0.0817
1.00	0.0361
Average precision over all relevant docs	
non-interpolated	0.2633

Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.4500
At 15 docs	0.4107
At 20 docs	0.3680
At 30 docs	0.3313
At 100 docs	0.2252
At 200 docs	0.1598
At 500 docs	0.0947
At 1000 docs	0.0574
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3000

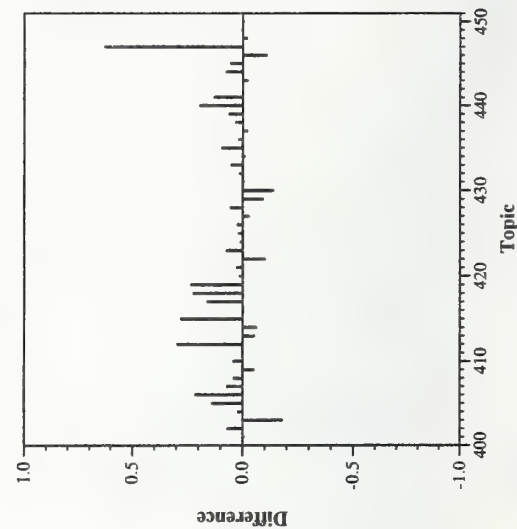
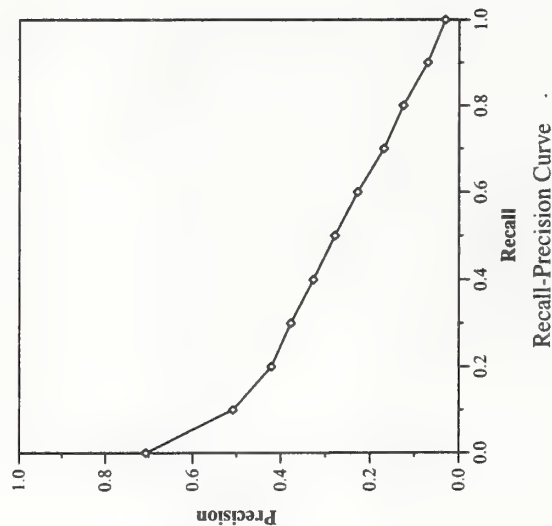


Ad hoc results — RICOH Co., Ltd.

Summary Statistics		
Run Number	ric8dpx	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3027	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7069
0.10	0.5080
0.20	0.4211
0.30	0.3778
0.40	0.3272
0.50	0.2789
0.60	0.2278
0.70	0.1693
0.80	0.1253
0.90	0.0703
1.00	0.0308
Average precision over all relevant docs	
non-interpolated	0.2748

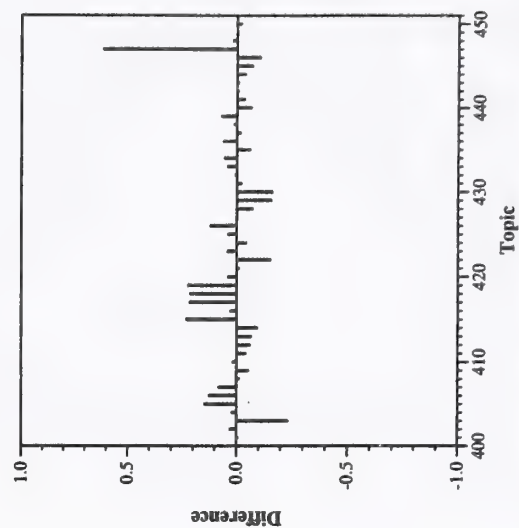
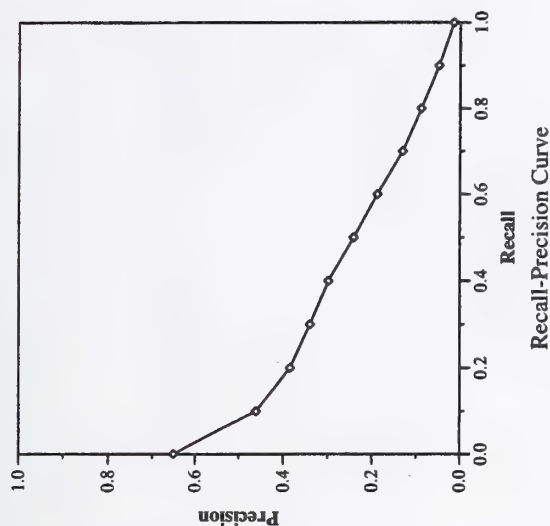
Document Level Averages	
	Precision
At 5 docs	0.5040
At 10 docs	0.4580
At 15 docs	0.4133
At 20 docs	0.3890
At 30 docs	0.3493
At 100 docs	0.2304
At 200 docs	0.1679
At 500 docs	0.0992
At 1000 docs	0.0605
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3074



Summary Statistics		
Run Number	ric8dnx	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2890	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6507
0.10	0.4607
0.20	0.3852
0.30	0.3400
0.40	0.2981
0.50	0.2411
0.60	0.1881
0.70	0.1308
0.80	0.0885
0.90	0.0482
1.00	0.0147
Average precision over all relevant docs	
non-interpolated	0.2426

Document Level Averages	
	Precision
At 5 docs	0.4800
At 10 docs	0.4380
At 15 docs	0.3920
At 20 docs	0.3520
At 30 docs	0.3160
At 100 docs	0.2076
At 200 docs	0.1554
At 500 docs	0.0925
At 1000 docs	0.0578
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2811

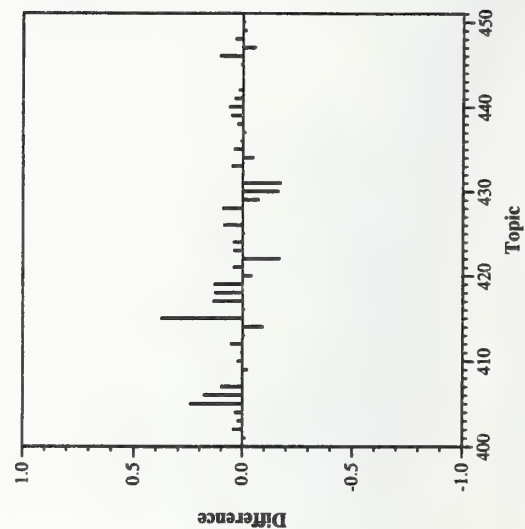
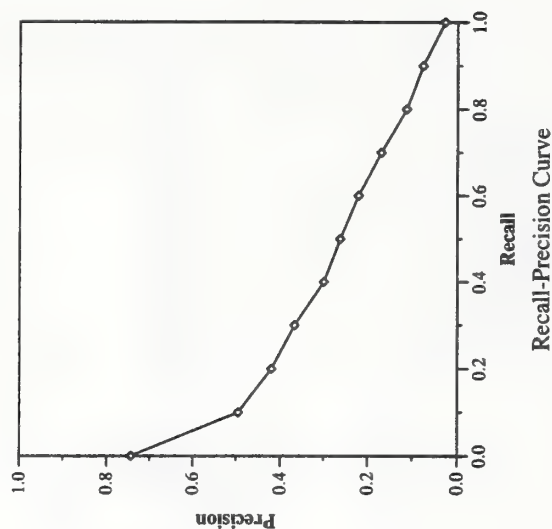


Ad hoc results — RICOH Co., Ltd.

Summary Statistics		
Run Number	ric8tpx	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3096	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7433
0.10	0.4952
0.20	0.4199
0.30	0.3675
0.40	0.2994
0.50	0.2618
0.60	0.2208
0.70	0.1697
0.80	0.1128
0.90	0.0760
1.00	0.0276
Average precision over all relevant docs	
non-interpolated	0.2689

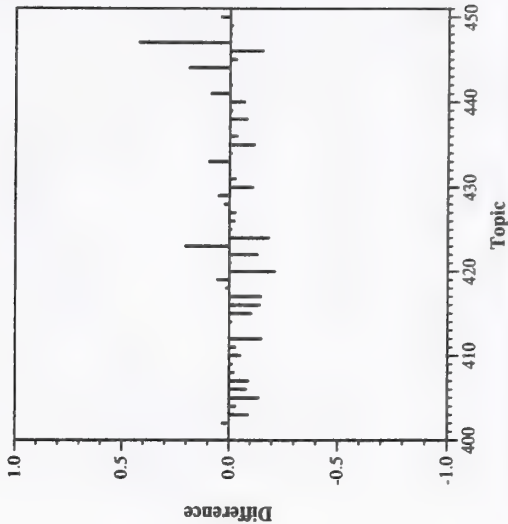
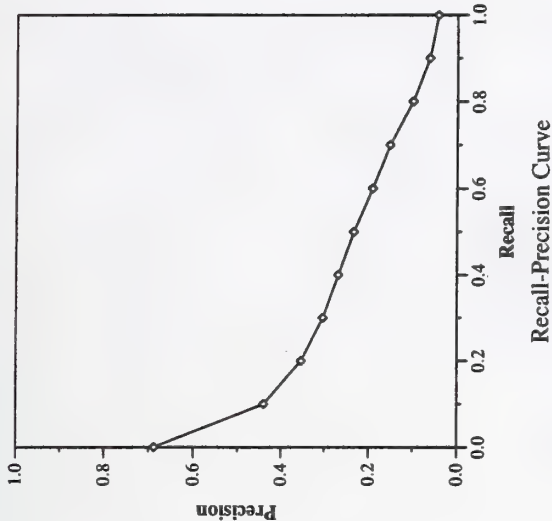
Document Level Averages	
	Precision
At 5 docs	0.5000
At 10 docs	0.4560
At 15 docs	0.4213
At 20 docs	0.3950
At 30 docs	0.3660
At 100 docs	0.2440
At 200 docs	0.1756
At 500 docs	0.1018
At 1000 docs	0.0619
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3007



Summary Statistics		
Run Number	mds08a3	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2640	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6880
0.10	0.4401
0.20	0.3531
0.30	0.3038
0.40	0.2685
0.50	0.2338
0.60	0.1913
0.70	0.1523
0.80	0.1001
0.90	0.0627
1.00	0.0434
Average precision over all relevant docs	
non-interpolated	0.2338

Document Level Averages	
	Precision
At 5 docs	0.4400
At 10 docs	0.4000
At 15 docs	0.3880
At 20 docs	0.3570
At 30 docs	0.3040
At 100 docs	0.1906
At 200 docs	0.1359
At 500 docs	0.0833
At 1000 docs	0.0528
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2793

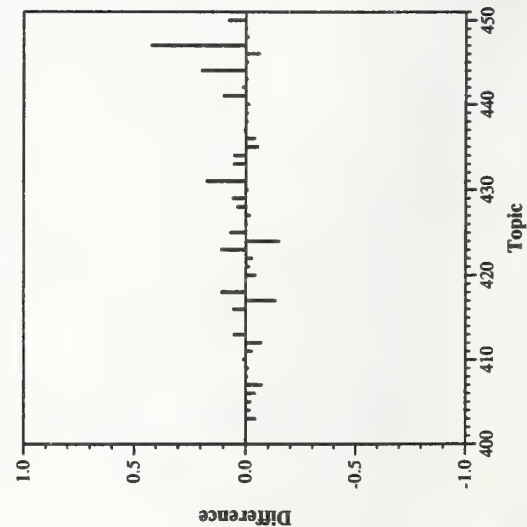
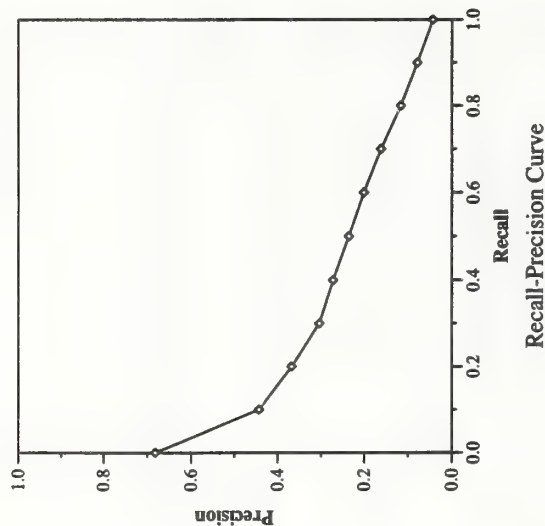


Ad hoc results — RMIT

Summary Statistics		
Run Number	mds08a2	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2732	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6817
0.10	0.4428
0.20	0.3676
0.30	0.3037
0.40	0.2717
0.50	0.2355
0.60	0.2010
0.70	0.1616
0.80	0.1161
0.90	0.0783
1.00	0.0431
Average precision over all relevant docs	
non-interpolated	0.2397

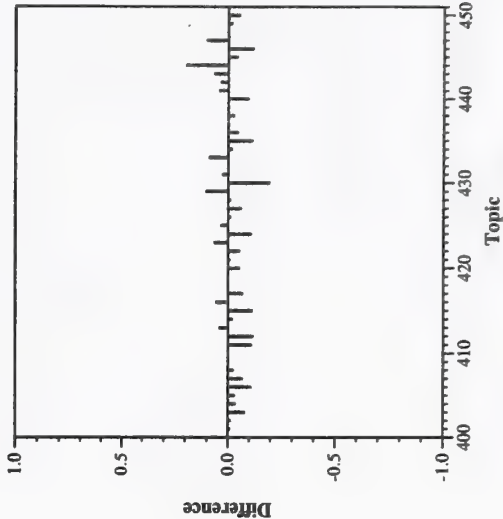
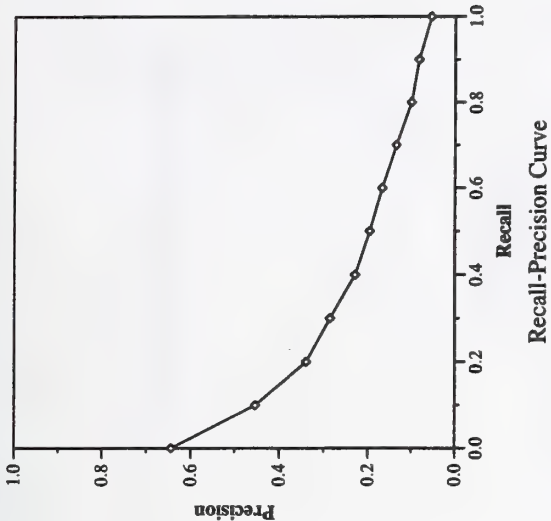
Document Level Averages	
	Precision
At 5 docs	0.4640
At 10 docs	0.4160
At 15 docs	0.3867
At 20 docs	0.3590
At 30 docs	0.3113
At 100 docs	0.2026
At 200 docs	0.1465
At 500 docs	0.0874
At 1000 docs	0.0546
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2890



Summary Statistics		
Run Number	mds08a1	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	48261	
Relevant:	4728	
Rel-ret:	2594	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6441
0.10	0.4540
0.20	0.3394
0.30	0.2855
0.40	0.2297
0.50	0.1964
0.60	0.1679
0.70	0.1353
0.80	0.1003
0.90	0.0831
1.00	0.0547
Average precision over all relevant docs	
non-interpolated	0.2236

Document Level Averages	
At 5 docs	0.4320
At 10 docs	0.4040
At 15 docs	0.3800
At 20 docs	0.3490
At 30 docs	0.3053
At 100 docs	0.2052
At 200 docs	0.1461
At 500 docs	0.0842
At 1000 docs	0.0519
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2693

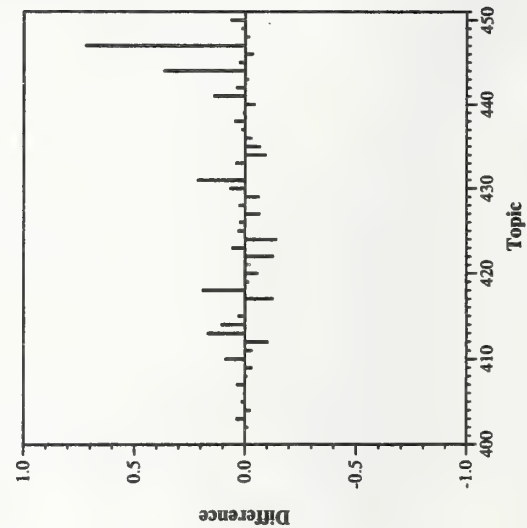
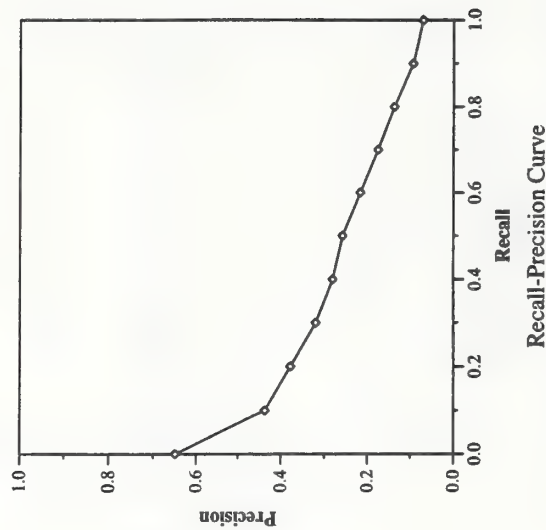


Ad hoc results — RMIT

Summary Statistics		
Run Number	mds08a4	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2843	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6494
0.10	0.4368
0.20	0.3780
0.30	0.3197
0.40	0.2813
0.50	0.2583
0.60	0.2172
0.70	0.1754
0.80	0.1374
0.90	0.0935
1.00	0.0702
Average precision over all relevant docs	
non-interpolated	0.2550

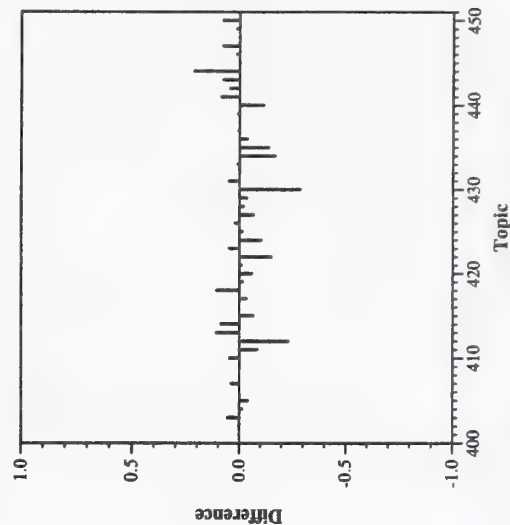
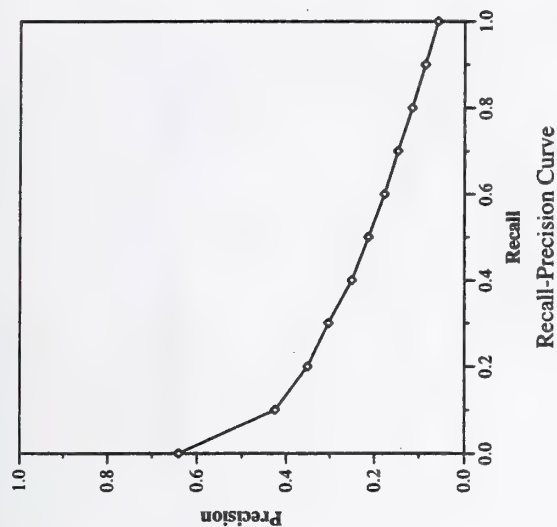
Document Level Averages	
	Precision
At 5 docs	0.4560
At 10 docs	0.4180
At 15 docs	0.4080
At 20 docs	0.3680
At 30 docs	0.3187
At 100 docs	0.1986
At 200 docs	0.1432
At 500 docs	0.0890
At 1000 docs	0.0569
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2894



Summary Statistics	
Run Number	mds08a5
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	2804

Recall Level Precision Averages	
Recall	Precision
0.00	0.6409
0.10	0.4254
0.20	0.3511
0.30	0.3038
0.40	0.2510
0.50	0.2135
0.60	0.1778
0.70	0.1477
0.80	0.1161
0.90	0.0873
1.00	0.0592
Average precision over all relevant docs	
non-interpolated	0.2324

Document Level Averages	
	Precision
At 5 docs	0.4360
At 10 docs	0.3900
At 15 docs	0.3653
At 20 docs	0.3470
At 30 docs	0.3107
At 100 docs	0.2076
At 200 docs	0.1462
At 500 docs	0.0874
At 1000 docs	0.0561
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2702

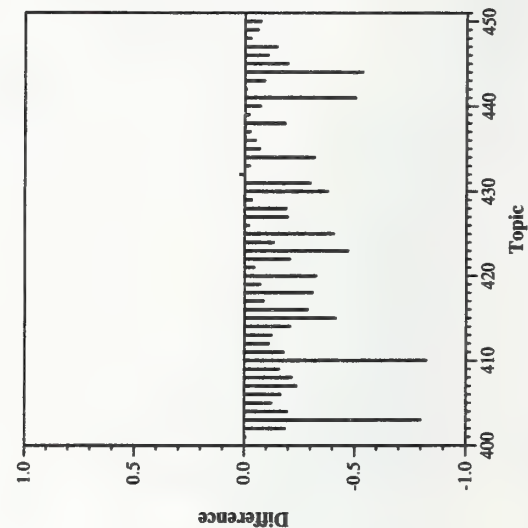
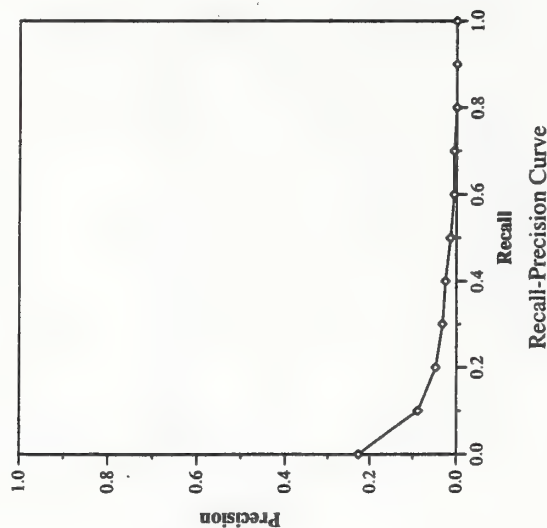


Ad hoc results — Rutgers University

Summary Statistics		
Run Number	Anthoc1	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	49317	
Relevant:	4728	
Rel-ret:	597	

Recall Level Precision Averages	
Recall	Precision
0.00	0.2274
0.10	0.0884
0.20	0.0481
0.30	0.0321
0.40	0.0254
0.50	0.0136
0.60	0.0058
0.70	0.0057
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0287

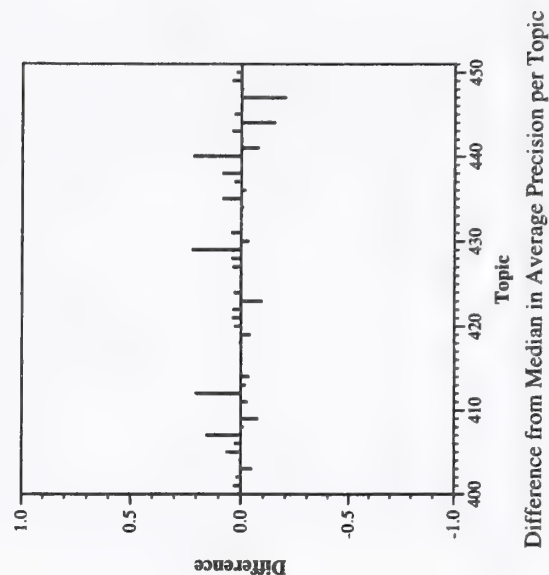
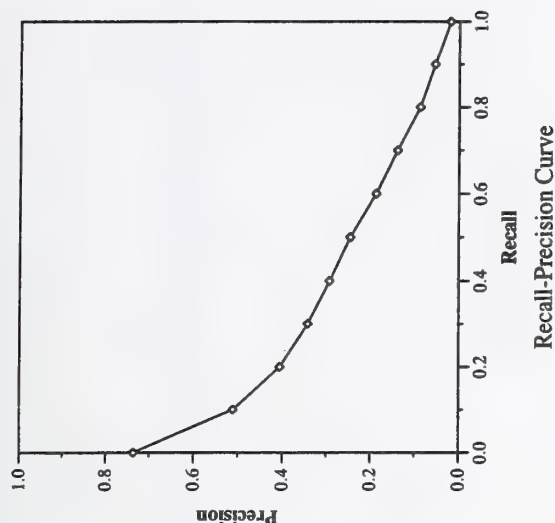
Document Level Averages	
	Precision
At 5 docs	0.0960
At 10 docs	0.0980
At 15 docs	0.0813
At 20 docs	0.0690
At 30 docs	0.0600
At 100 docs	0.0376
At 200 docs	0.0266
At 500 docs	0.0166
At 1000 docs	0.0119
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0583



Summary Statistics	
Run Number	Sab8A1
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3006

Recall Level Precision Averages	
Recall	Precision
0.00	0.7360
0.10	0.5107
0.20	0.4059
0.30	0.3424
0.40	0.2931
0.50	0.2457
0.60	0.1873
0.70	0.1391
0.80	0.0881
0.90	0.0545
1.00	0.0197
Average precision over all relevant docs	
non-interpolated	0.2553

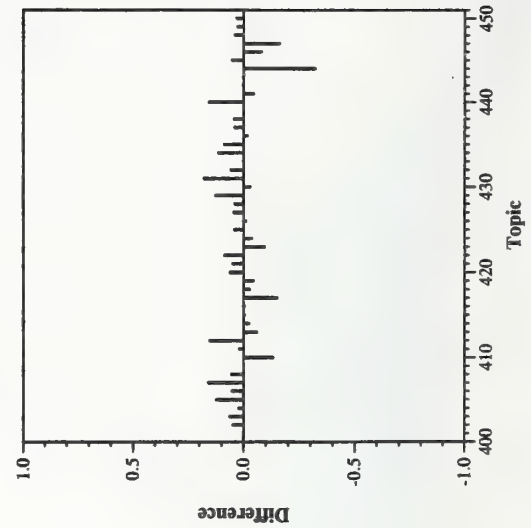
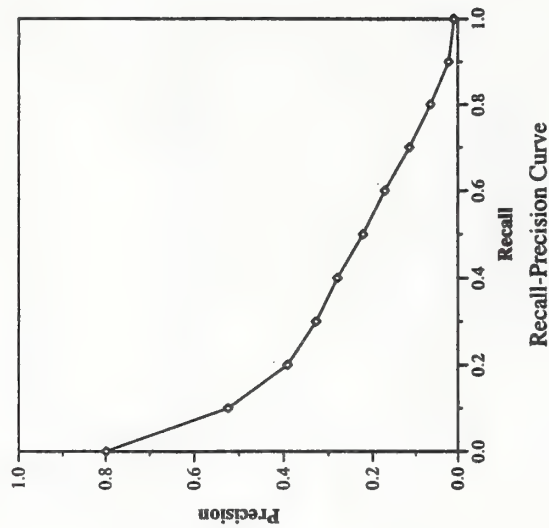
Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.4780
At 15 docs	0.4387
At 20 docs	0.4000
At 30 docs	0.3687
At 100 docs	0.2376
At 200 docs	0.1683
At 500 docs	0.0980
At 1000 docs	0.0601
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2901



Summary Statistics		
Run Number	Sab8A2	
Run Description	Automatic, desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2829	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7988
0.10	0.5247
0.20	0.3913
0.30	0.3277
0.40	0.2802
0.50	0.2222
0.60	0.1724
0.70	0.1149
0.80	0.0655
0.90	0.0219
1.00	0.0094
Average precision over all relevant docs	
non-interpolated	0.2407

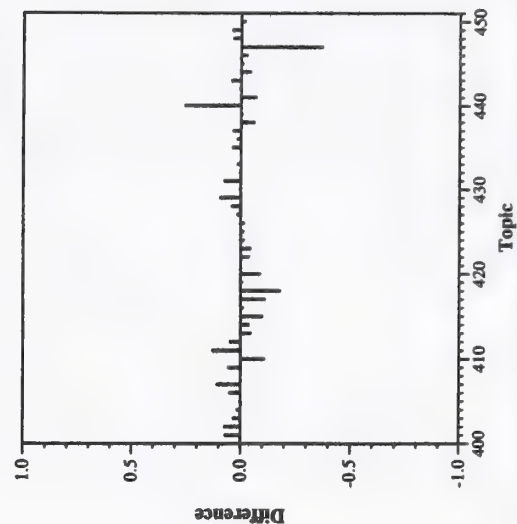
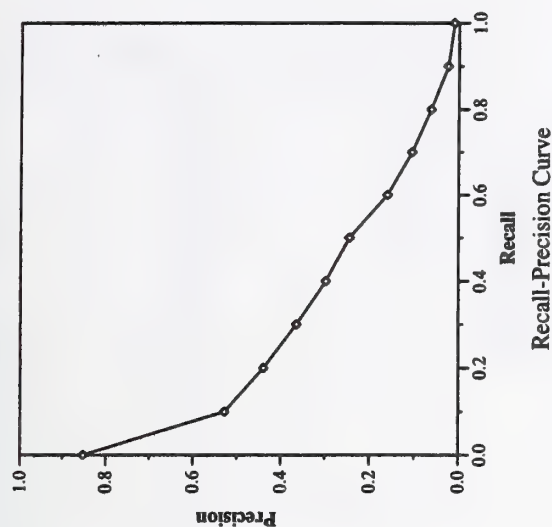
Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.4700
At 15 docs	0.4147
At 20 docs	0.3830
At 30 docs	0.3407
At 100 docs	0.2228
At 200 docs	0.1603
At 500 docs	0.0936
At 1000 docs	0.0566
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2829



Summary Statistics		
Run Number	Sab8A3	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2957	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8514
0.10	0.5286
0.20	0.4400
0.30	0.3645
0.40	0.2983
0.50	0.2453
0.60	0.1603
0.70	0.1054
0.80	0.0629
0.90	0.0246
1.00	0.0108
Average precision over all relevant docs	
non-interpolated	0.2546

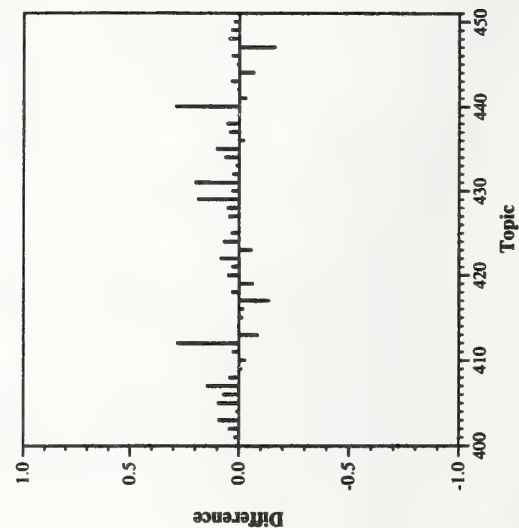
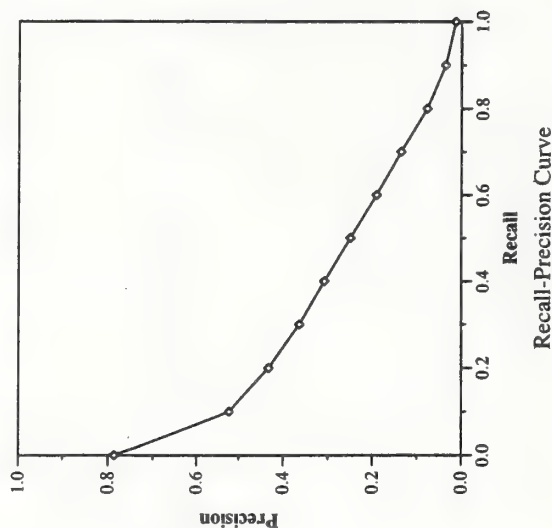
Document Level Averages	
	Precision
At 5 docs	0.5600
At 10 docs	0.4940
At 15 docs	0.4400
At 20 docs	0.4000
At 30 docs	0.3567
At 100 docs	0.2316
At 200 docs	0.1633
At 500 docs	0.0968
At 1000 docs	0.0591
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3088



Summary Statistics	
Run Number	Sab8A4
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	2986

Recall Level Precision Averages	
Recall	Precision
0.00	0.7860
0.10	0.5229
0.20	0.4324
0.30	0.3644
0.40	0.3084
0.50	0.2498
0.60	0.1912
0.70	0.1360
0.80	0.0776
0.90	0.0362
1.00	0.0133
Average precision over all relevant docs	
non-interpolated	0.2608

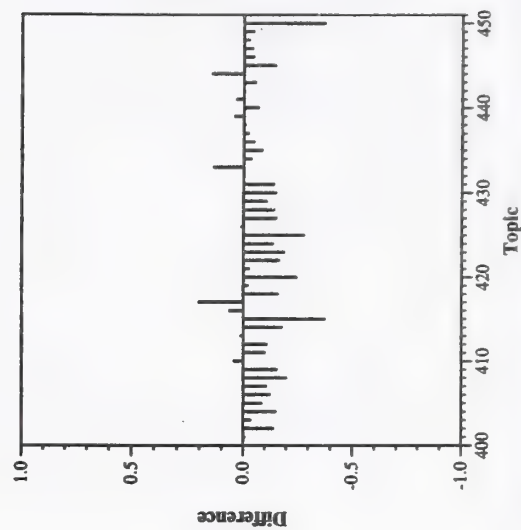
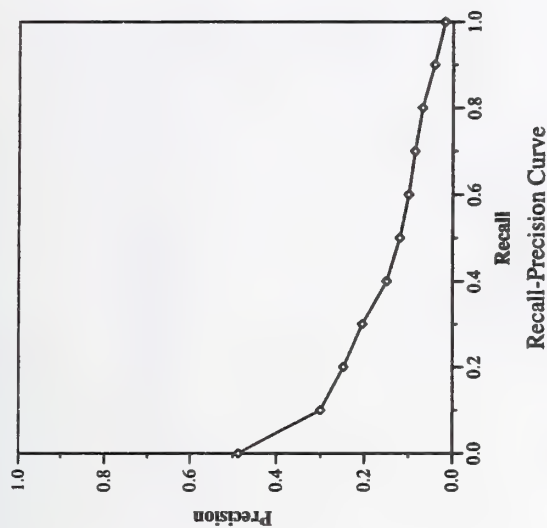
Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4800
At 15 docs	0.4413
At 20 docs	0.4090
At 30 docs	0.3733
At 100 docs	0.2384
At 200 docs	0.1702
At 500 docs	0.0985
At 1000 docs	0.0597
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3021



Summary Statistics		
Run Number	Scai8Adhoc	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	47182	
Relevant:	4728	
Rel-ret:	1914	

Recall Level Precision Averages	
Recall	Precision
0.00	0.4873
0.10	0.3007
0.20	0.2478
0.30	0.2045
0.40	0.1491
0.50	0.1197
0.60	0.0998
0.70	0.0858
0.80	0.0690
0.90	0.0415
1.00	0.0180
Average precision over all relevant docs	
non-interpolated	0.1461

Document Level Averages	
	Precision
At 5 docs	0.2880
At 10 docs	0.2520
At 15 docs	0.2320
At 20 docs	0.2260
At 30 docs	0.2047
At 100 docs	0.1364
At 200 docs	0.0958
At 500 docs	0.0568
At 1000 docs	0.0383
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1908

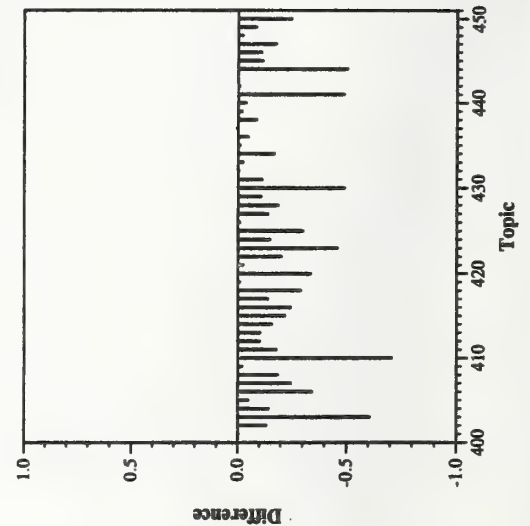
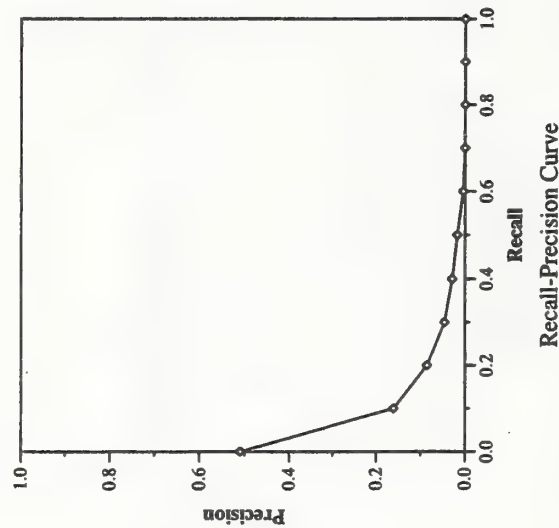


Ad hoc results — State University of New York at Buffalo

Summary Statistics		
Run Number	UB99SW	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	1430	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5099
0.10	0.1611
0.20	0.0856
0.30	0.0457
0.40	0.0285
0.50	0.0173
0.60	0.0044
0.70	0.0007
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0550

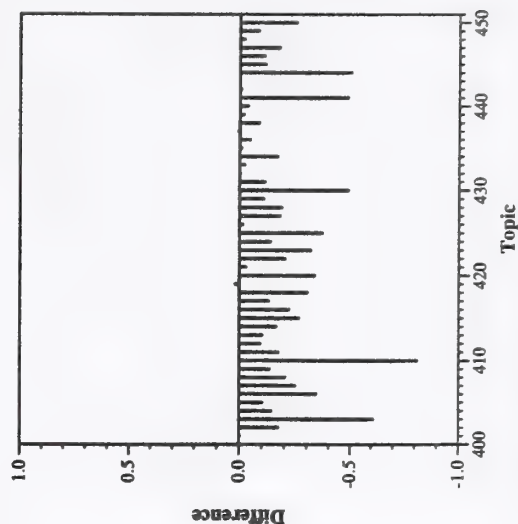
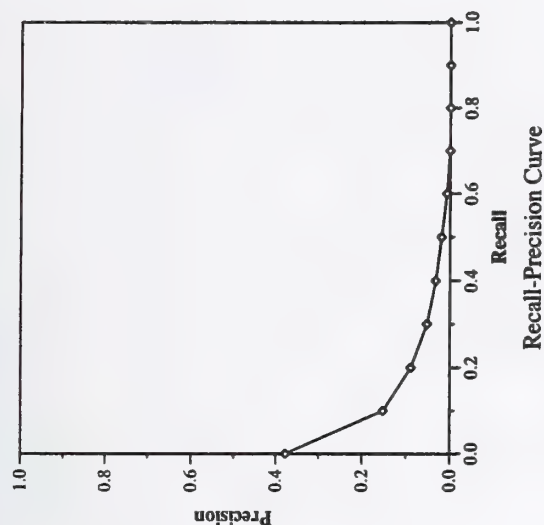
Document Level Averages	
	Precision
At 5 docs	0.2400
At 10 docs	0.1900
At 15 docs	0.1680
At 20 docs	0.1480
At 30 docs	0.1240
At 100 docs	0.0834
At 200 docs	0.0618
At 500 docs	0.0418
At 1000 docs	0.0286
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0945



Summary Statistics		
Run Number	UB99T	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	1432	

Recall Level Precision Averages	
Recall	Precision
0.00	0.3776
0.10	0.1517
0.20	0.0884
0.30	0.0521
0.40	0.0322
0.50	0.0189
0.60	0.0071
0.70	0.0005
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0493

Document Level Averages	
	Precision
At 5 docs	0.1560
At 10 docs	0.1460
At 15 docs	0.1413
At 20 docs	0.1340
At 30 docs	0.1227
At 100 docs	0.0778
At 200 docs	0.0577
At 500 docs	0.0413
At 1000 docs	0.0286
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0940

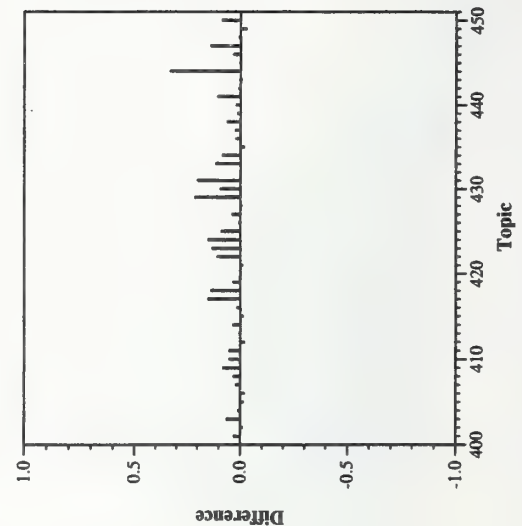
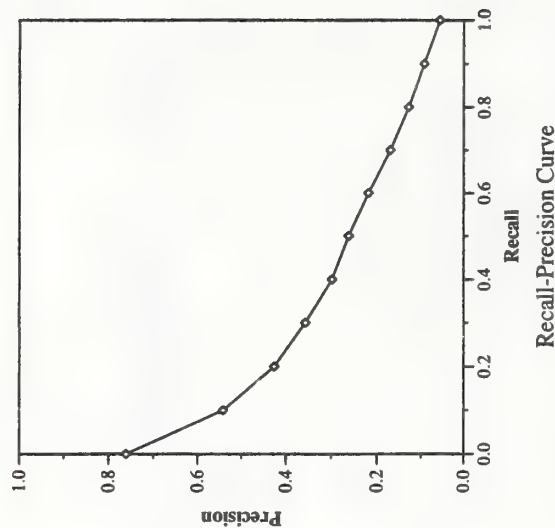


Ad hoc results — TwentyOne

Summary Statistics		
Run Number	tno8d4	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3063	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7614
0.10	0.5419
0.20	0.4262
0.30	0.3570
0.40	0.2986
0.50	0.2623
0.60	0.2193
0.70	0.1699
0.80	0.1284
0.90	0.0925
1.00	0.0563
Average precision over all relevant docs	
non-interpolated	0.2778

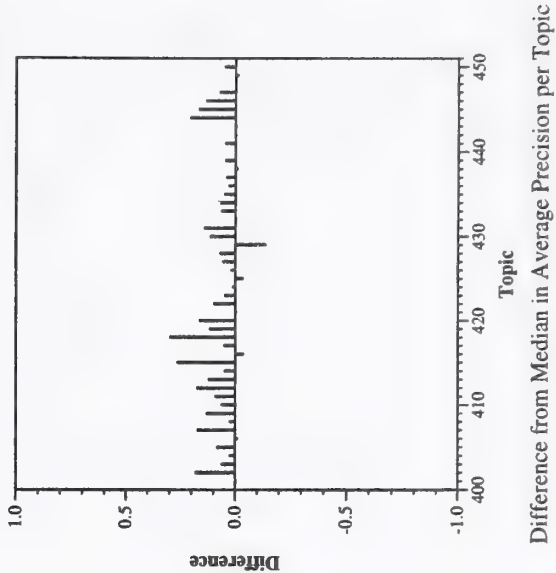
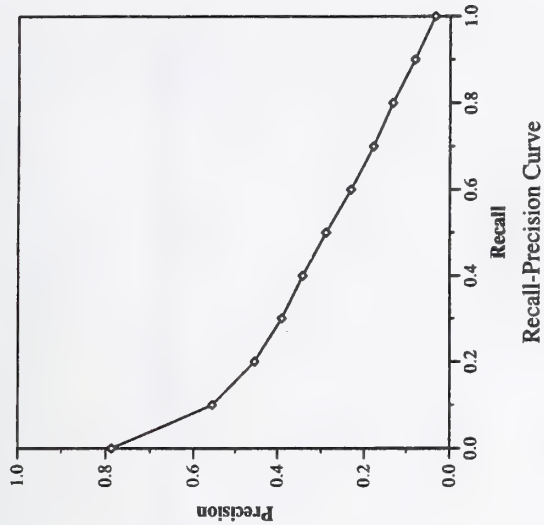
Document Level Averages	
	Precision
At 5 docs	0.5480
At 10 docs	0.4880
At 15 docs	0.4613
At 20 docs	0.4230
At 30 docs	0.3900
At 100 docs	0.2338
At 200 docs	0.1666
At 500 docs	0.0987
At 1000 docs	0.0613
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3165



Summary Statistics		
Run Number	tno8d3	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3248	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7872
0.10	0.5555
0.20	0.4541
0.30	0.3905
0.40	0.3427
0.50	0.2891
0.60	0.2314
0.70	0.1787
0.80	0.1329
0.90	0.0809
1.00	0.0340
Average precision over all relevant docs	
non-interpolated	0.2921

Document Level Averages	
At 5 docs	0.5000
At 10 docs	0.4540
At 15 docs	0.4333
At 20 docs	0.4090
At 30 docs	0.3680
At 100 docs	0.2506
At 200 docs	0.1812
At 500 docs	0.1061
At 1000 docs	0.0650
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3161

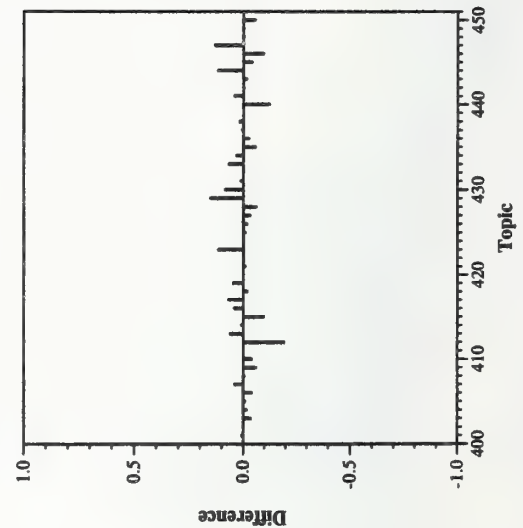
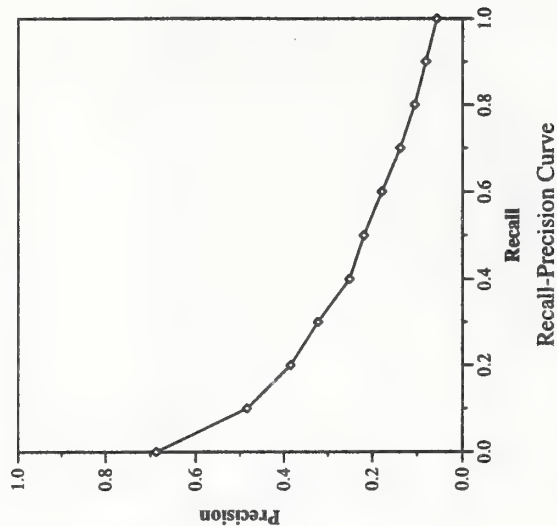


Ad hoc results — TwentyOne

Summary Statistics	
Run Number	tno8t2
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49085
Relevant:	4728
Rel-ret:	2741

Recall Level Precision Averages	
Recall	Precision
0.00	0.6873
0.10	0.4841
0.20	0.3849
0.30	0.3224
0.40	0.2513
0.50	0.2191
0.60	0.1788
0.70	0.1379
0.80	0.1055
0.90	0.0810
1.00	0.0571
Average precision over all relevant docs	
non-interpolated	0.2423

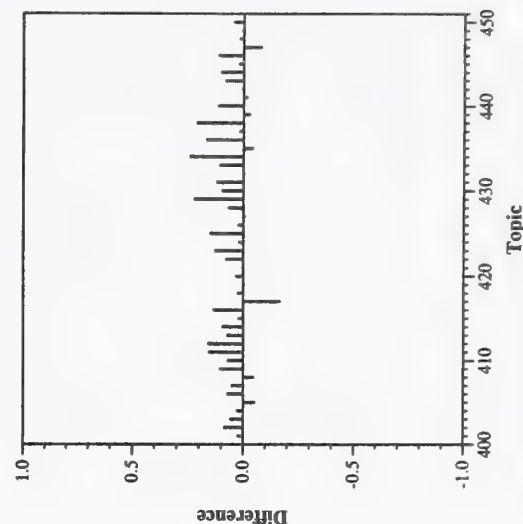
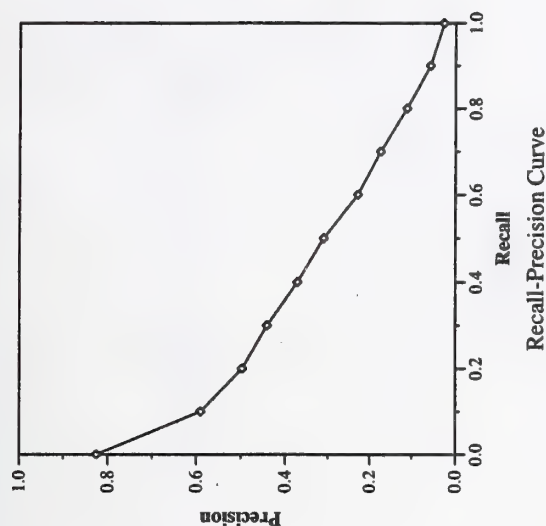
Document Level Averages	
	Precision
At 5 docs	0.4520
At 10 docs	0.4300
At 15 docs	0.3867
At 20 docs	0.3700
At 30 docs	0.3380
At 100 docs	0.2166
At 200 docs	0.1525
At 500 docs	0.0877
At 1000 docs	0.0548
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2928



Summary Statistics		
Run Number	UniNET8Lg	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3168	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8256
0.10	0.5901
0.20	0.4976
0.30	0.4416
0.40	0.3719
0.50	0.3104
0.60	0.2290
0.70	0.1739
0.80	0.1120
0.90	0.0579
1.00	0.0266
Average precision over all relevant docs	
non-interpolated	0.3138

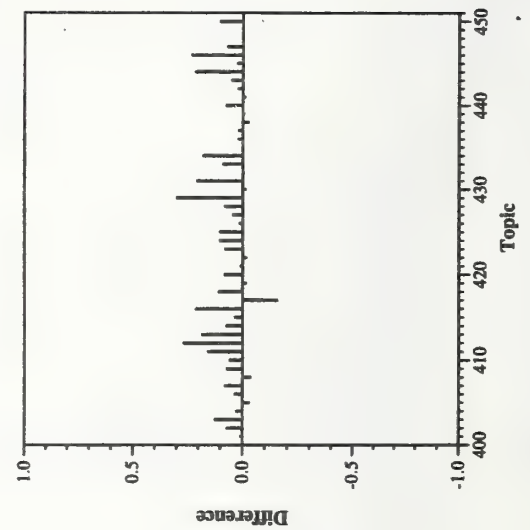
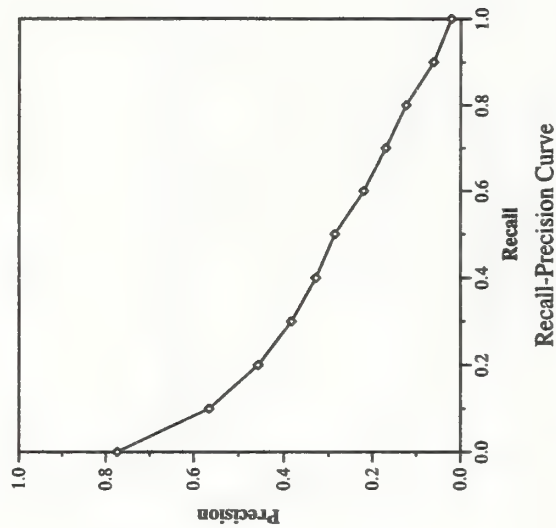
Document Level Averages	
	Precision
At 5 docs	0.6120
At 10 docs	0.5560
At 15 docs	0.5027
At 20 docs	0.4570
At 30 docs	0.4113
At 100 docs	0.2674
At 200 docs	0.1890
At 500 docs	0.1068
At 1000 docs	0.0634
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3443



Summary Statistics		
Run Number	UniNET8St	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2977	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7740
0.10	0.5649
0.20	0.4569
0.30	0.3832
0.40	0.3288
0.50	0.2862
0.60	0.2201
0.70	0.1699
0.80	0.1234
0.90	0.0610
1.00	0.0215
Average precision over all relevant docs	
non-interpolated	0.2906

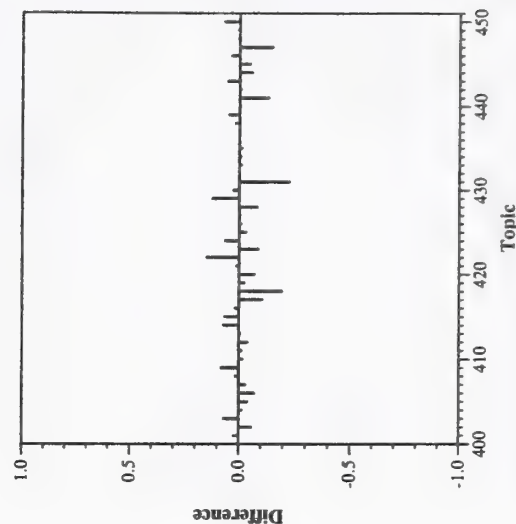
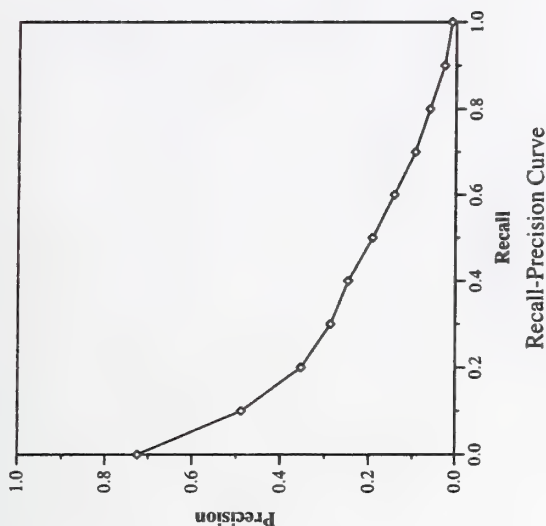
Document Level Averages	
	Precision
At 5 docs	0.5480
At 10 docs	0.5000
At 15 docs	0.4707
At 20 docs	0.4370
At 30 docs	0.3820
At 100 docs	0.2460
At 200 docs	0.1755
At 500 docs	0.1004
At 1000 docs	0.0595
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3235



Summary Statistics		
Run Number	umd99a1	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2660	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7250
0.10	0.4886
0.20	0.3535
0.30	0.2873
0.40	0.2476
0.50	0.1922
0.60	0.1430
0.70	0.0948
0.80	0.0613
0.90	0.0275
1.00	0.0102
Average precision over all relevant docs	
non-interpolated	0.2141

Document Level Averages	
	Precision
At 5 docs	0.5000
At 10 docs	0.4340
At 15 docs	0.3960
At 20 docs	0.3710
At 30 docs	0.3273
At 100 docs	0.2054
At 200 docs	0.1439
At 500 docs	0.0843
At 1000 docs	0.0532
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2658

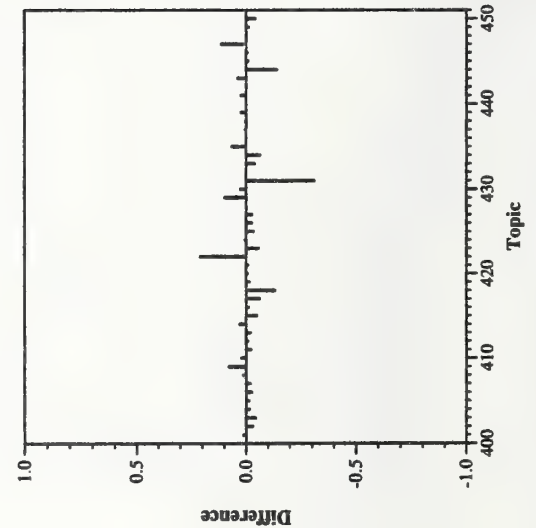
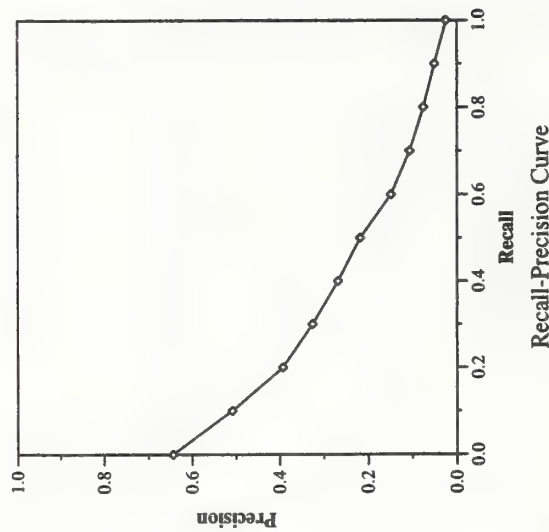


Ad hoc results — University of Massachusetts

Summary Statistics	
Run Number	INQ601
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	2701

Recall Level Precision Averages	
Recall	Precision
0.00	0.6444
0.10	0.5087
0.20	0.3927
0.30	0.3256
0.40	0.2680
0.50	0.2181
0.60	0.1486
0.70	0.1068
0.80	0.0760
0.90	0.0510
1.00	0.0247
Average precision over all relevant docs	
non-interpolated	0.2325

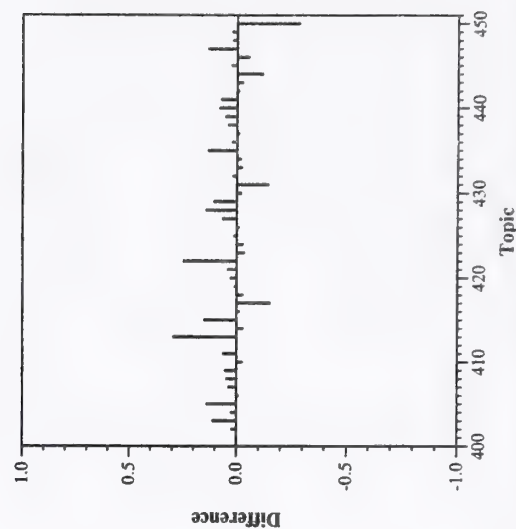
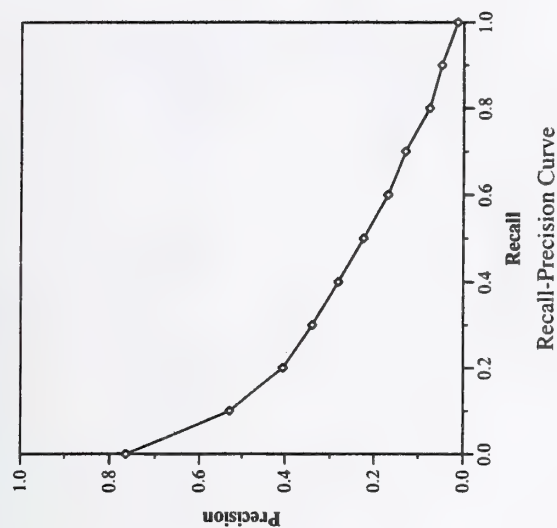
Document Level Averages	
	Precision
At 5 docs	0.4640
At 10 docs	0.4360
At 15 docs	0.4013
At 20 docs	0.3730
At 30 docs	0.3353
At 100 docs	0.2230
At 200 docs	0.1549
At 500 docs	0.0872
At 1000 docs	0.0540
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2824



Summary Statistics	
Run Number	INQ602
Run Description	Automatic, desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	2699

Recall Level Precision Averages	
Recall	Precision
0.00	0.7642
0.10	0.5298
0.20	0.4053
0.30	0.3399
0.40	0.2815
0.50	0.2240
0.60	0.1691
0.70	0.1298
0.80	0.0761
0.90	0.0489
1.00	0.0142
Average precision over all relevant docs	
non-interpolated	0.2492

Document Level Averages	
	Precision
At 5 docs	0.4960
At 10 docs	0.4720
At 15 docs	0.4280
At 20 docs	0.3960
At 30 docs	0.3533
At 100 docs	0.2196
At 200 docs	0.1559
At 500 docs	0.0888
At 1000 docs	0.0540
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2898

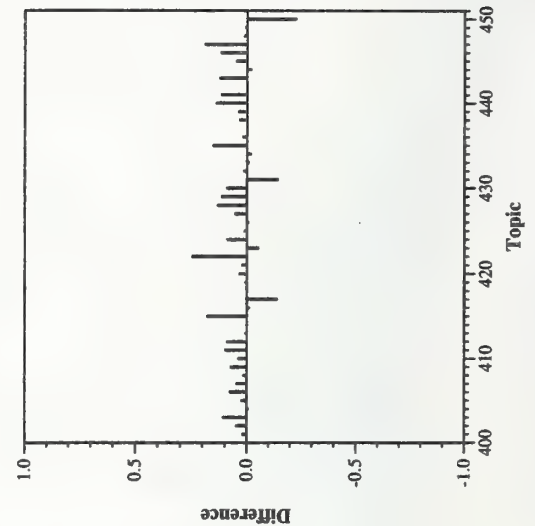
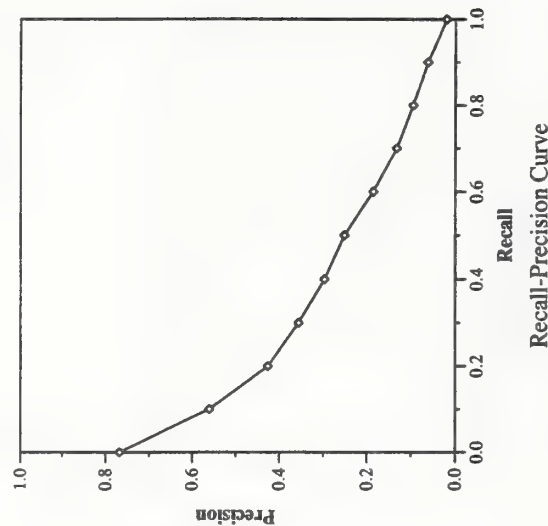


Ad hoc results — University of Massachusetts

Summary Statistics		
Run Number	INQ603	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2907	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7687
0.10	0.5605
0.20	0.4262
0.30	0.3562
0.40	0.2975
0.50	0.2520
0.60	0.1859
0.70	0.1322
0.80	0.0956
0.90	0.0616
1.00	0.0195
Average precision over all relevant docs	
non-interpolated	0.2659

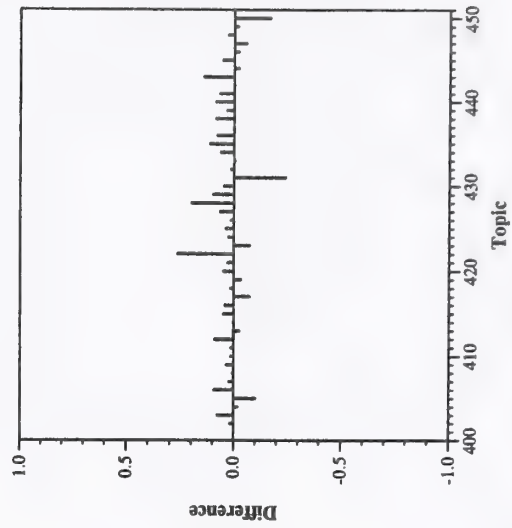
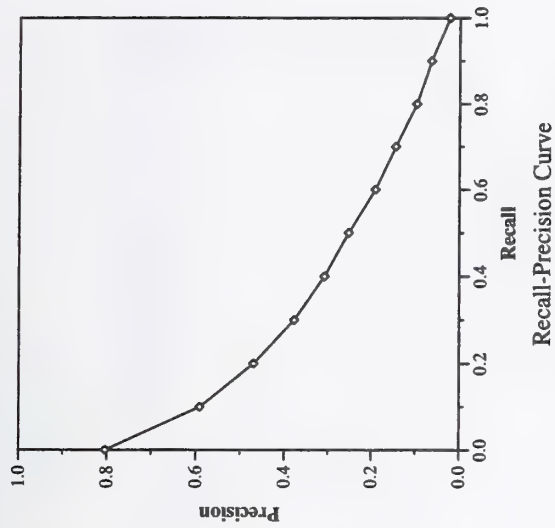
Document Level Averages	
	Precision
At 5 docs	0.5320
At 10 docs	0.4760
At 15 docs	0.4387
At 20 docs	0.4080
At 30 docs	0.3633
At 100 docs	0.2382
At 200 docs	0.1674
At 500 docs	0.0966
At 1000 docs	0.0581
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3048



Summary Statistics		
Run Number	INQ604	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3098	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8037
0.10	0.5909
0.20	0.4699
0.30	0.3778
0.40	0.3084
0.50	0.2523
0.60	0.1913
0.70	0.1454
0.80	0.0979
0.90	0.0646
1.00	0.0230
Average precision over all relevant docs	
non-interpolated	0.2809

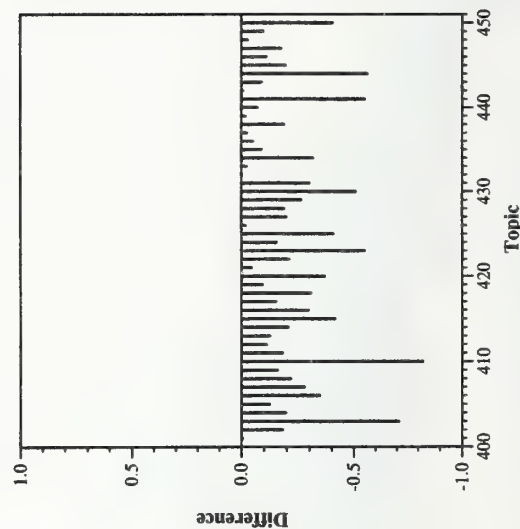
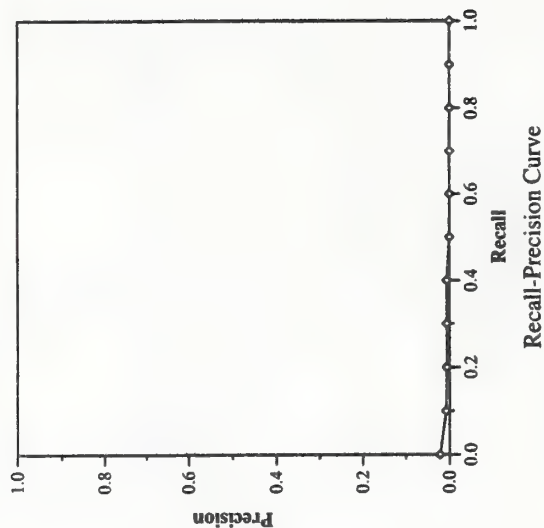
Document Level Averages	
	Precision
At 5 docs	0.5840
At 10 docs	0.5020
At 15 docs	0.4680
At 20 docs	0.4420
At 30 docs	0.3913
At 100 docs	0.2470
At 200 docs	0.1786
At 500 docs	0.1023
At 1000 docs	0.0620
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3162



Summary Statistics		
Run Number	isa25	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	90	

Recall Level Precision Averages	
Recall	Precision
0.00	0.0218
0.10	0.0078
0.20	0.0062
0.30	0.0062
0.40	0.0062
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0026

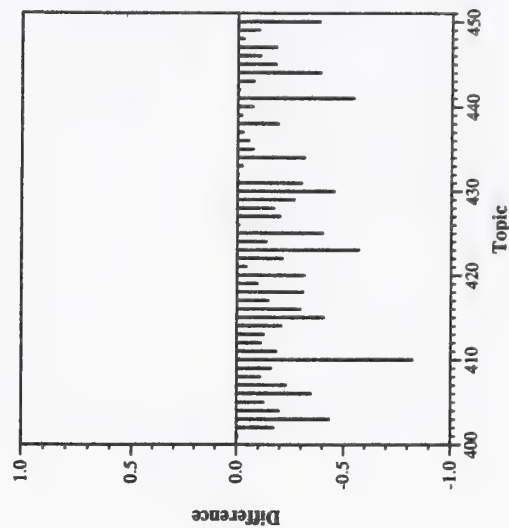
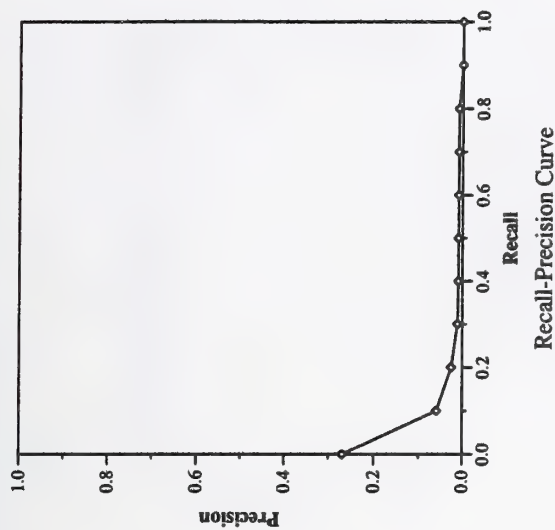
Document Level Averages	
	Precision
At 5 docs	0.0040
At 10 docs	0.0040
At 15 docs	0.0053
At 20 docs	0.0080
At 30 docs	0.0100
At 100 docs	0.0058
At 200 docs	0.0037
At 500 docs	0.0023
At 1000 docs	0.0018
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0081



Summary Statistics		
Run Number	isa50	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	49224	
Relevant:	4728	
Rel-ret:	441	

Recall Level Precision Averages	
Recall	Precision
0.00	0.2693
0.10	0.0593
0.20	0.0254
0.30	0.0112
0.40	0.0092
0.50	0.0092
0.60	0.0090
0.70	0.0087
0.80	0.0087
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0203

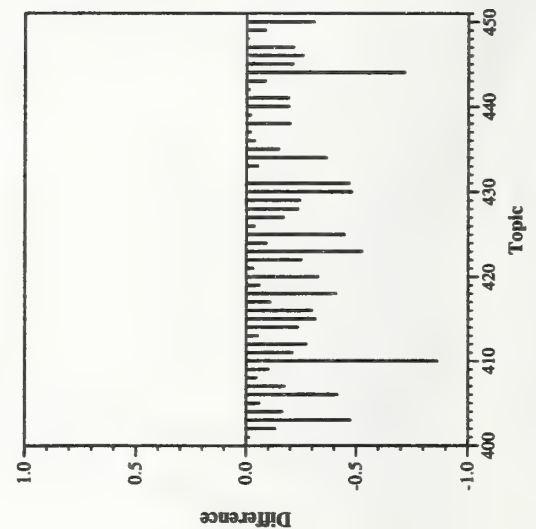
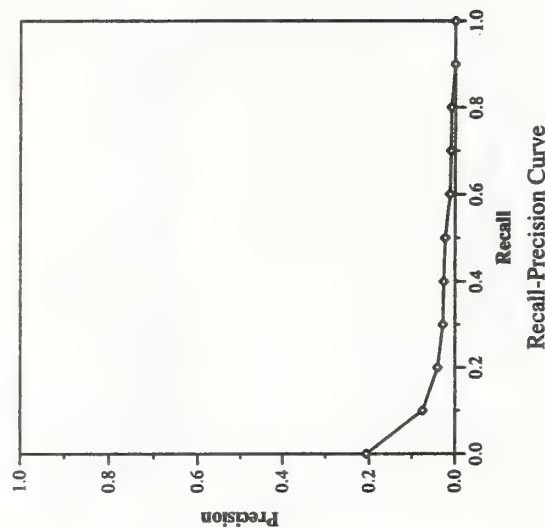
Document Level Averages	
	Precision
At 5 docs	0.1000
At 10 docs	0.0740
At 15 docs	0.0613
At 20 docs	0.0530
At 30 docs	0.0467
At 100 docs	0.0260
At 200 docs	0.0165
At 500 docs	0.0111
At 1000 docs	0.0088
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0349



Summary Statistics		
Run Number	isa25t	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	49065	
Relevant:	4728	
Rel-ret:	549	

Recall Level Precision Averages	
Recall	Precision
0.00	0.2066
0.10	0.0753
0.20	0.0403
0.30	0.0279
0.40	0.0260
0.50	0.0225
0.60	0.0120
0.70	0.0097
0.80	0.0087
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0273

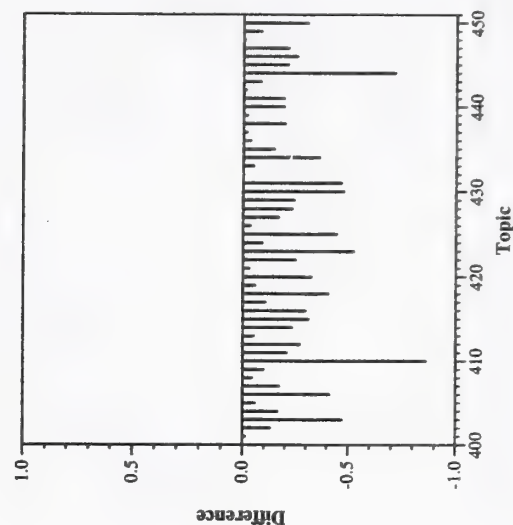
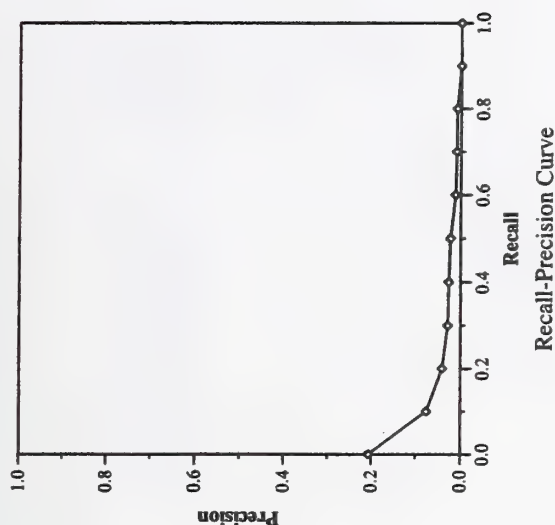
Document Level Averages	
	Precision
At 5 docs	0.0800
At 10 docs	0.0920
At 15 docs	0.0933
At 20 docs	0.0830
At 30 docs	0.0687
At 100 docs	0.0404
At 200 docs	0.0258
At 500 docs	0.0150
At 1000 docs	0.0110
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0515



Summary Statistics		
Run Number	isa50t	
Run Description	Automatic, title	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	49065	
Relevant:	4728	
Rel-ret:	549	

Recall Level Precision Averages	
Recall	Precision
0.00	0.2066
0.10	0.0753
0.20	0.0403
0.30	0.0279
0.40	0.0260
0.50	0.0225
0.60	0.0120
0.70	0.0097
0.80	0.0087
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0273

Document Level Averages	
	Precision
At 5 docs	0.0800
At 10 docs	0.0920
At 15 docs	0.0933
At 20 docs	0.0830
At 30 docs	0.0687
At 100 docs	0.0404
At 200 docs	0.0258
At 500 docs	0.0150
At 1000 docs	0.0110
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0515

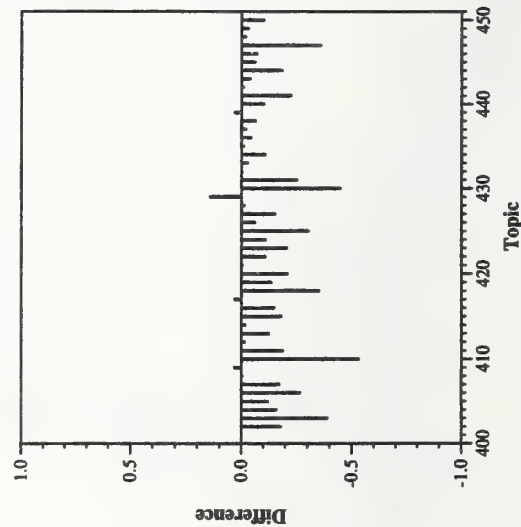
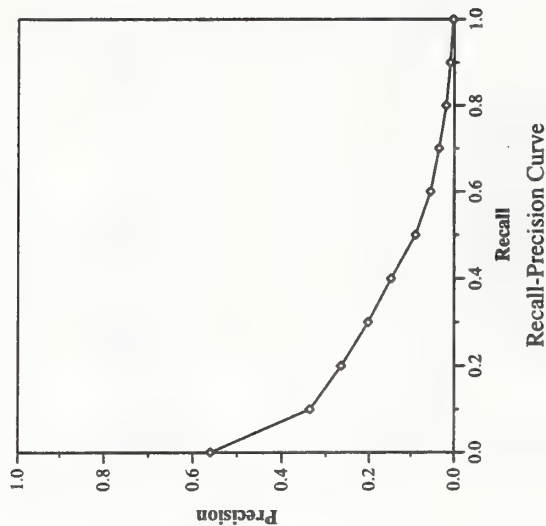


Ad hoc results — University of North Carolina (Yang)

Summary Statistics		
Run Number	unc8al32	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2249	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5602
0.10	0.3367
0.20	0.2653
0.30	0.2024
0.40	0.1482
0.50	0.0895
0.60	0.0545
0.70	0.0355
0.80	0.0188
0.90	0.0094
1.00	0.0029
Average precision over all relevant docs	
non-interpolated	0.1347

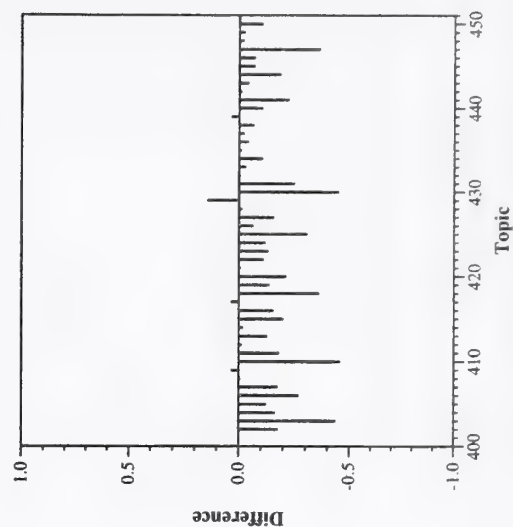
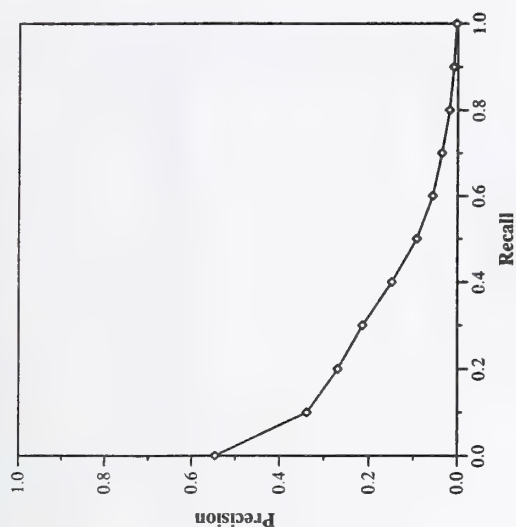
Document Level Averages	
	Precision
At 5 docs	0.3280
At 10 docs	0.3080
At 15 docs	0.2760
At 20 docs	0.2560
At 30 docs	0.2367
At 100 docs	0.1642
At 200 docs	0.1202
At 500 docs	0.0715
At 1000 docs	0.0450
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1934



Summary Statistics		
Run Number	unc8al42	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2261	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5460
0.10	0.3382
0.20	0.2693
0.30	0.2145
0.40	0.1485
0.50	0.0921
0.60	0.0565
0.70	0.0358
0.80	0.0186
0.90	0.0091
1.00	0.0029
Average precision over all relevant docs	
non-interpolated	0.1372

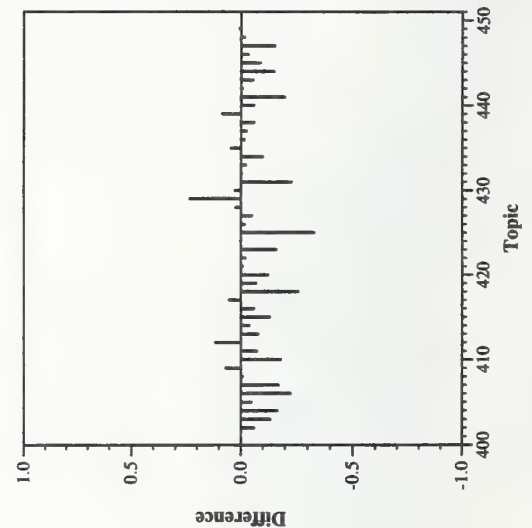
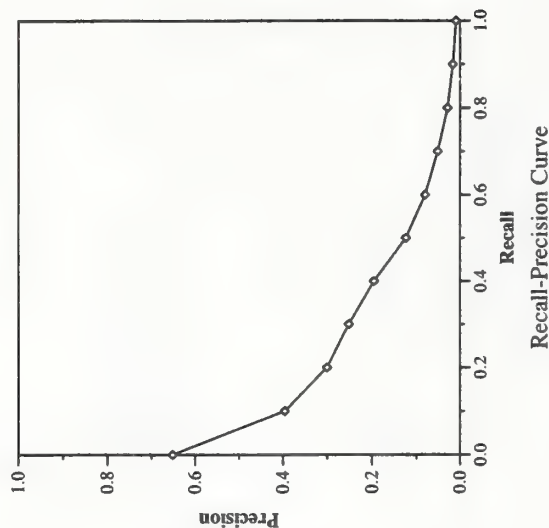
Document Level Averages	
	Precision
At 5 docs	0.3280
At 10 docs	0.3160
At 15 docs	0.2827
At 20 docs	0.2610
At 30 docs	0.2453
At 100 docs	0.1664
At 200 docs	0.1214
At 500 docs	0.0712
At 1000 docs	0.0452
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1972



Summary Statistics	
Run Number	unc8al52
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	2281

Recall Level Precision Averages	
Recall	Precision
0.00	0.6516
0.10	0.3960
0.20	0.3008
0.30	0.2519
0.40	0.1956
0.50	0.1238
0.60	0.0807
0.70	0.0518
0.80	0.0290
0.90	0.0167
1.00	0.0097
Average precision over all relevant docs	
non-interpolated	0.1669

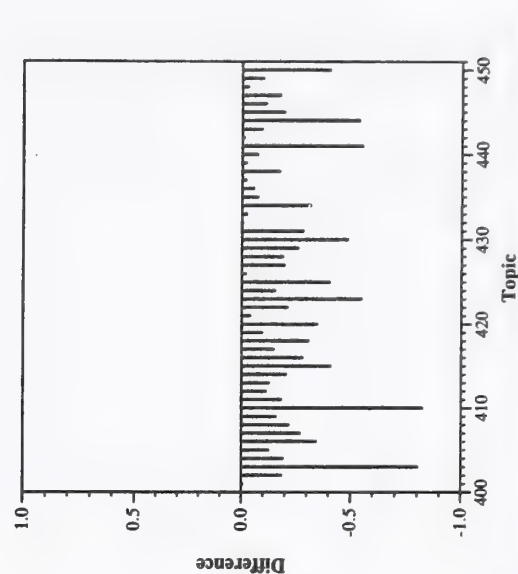
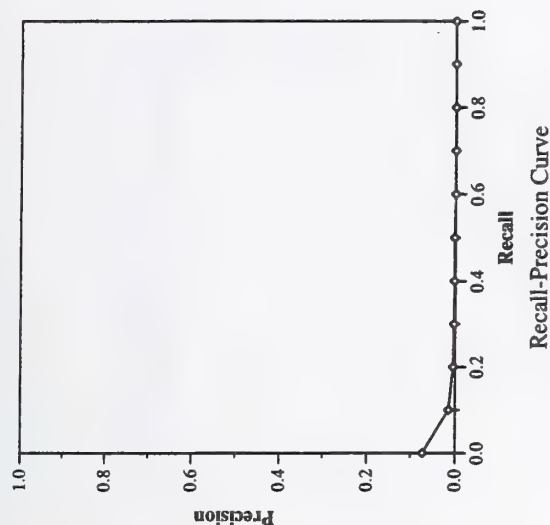
Document Level Averages	
	Precision
At 5 docs	0.4120
At 10 docs	0.3580
At 15 docs	0.3440
At 20 docs	0.3100
At 30 docs	0.2660
At 100 docs	0.1690
At 200 docs	0.1204
At 500 docs	0.0719
At 1000 docs	0.0456
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2172



Summary Statistics		
Run Number	1	
Run Description	Automatic, desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	31340	
Relevant:	4728.	
Rel-ret:	315	

Recall Level Precision Averages	
Recall	Precision
0.00	0.0726
0.10	0.0148
0.20	0.0045
0.30	0.0025
0.40	0.0018
0.50	0.0015
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0043

Document Level Averages	
	Precision
At 5 docs	0.0280
At 10 docs	0.0240
At 15 docs	0.0253
At 20 docs	0.0200
At 30 docs	0.0200
At 100 docs	0.0138
At 200 docs	0.0118
At 500 docs	0.0105
At 1000 docs	0.0063
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0137

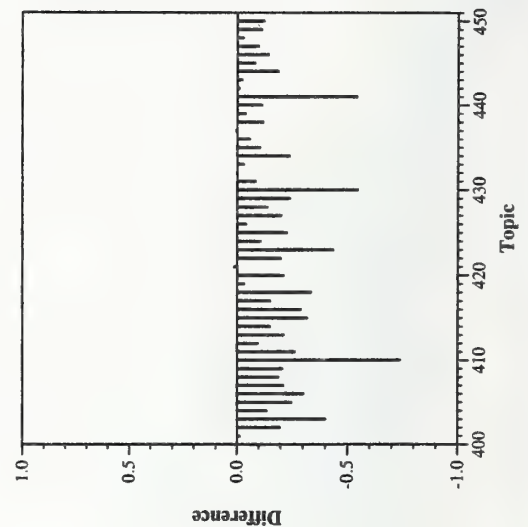
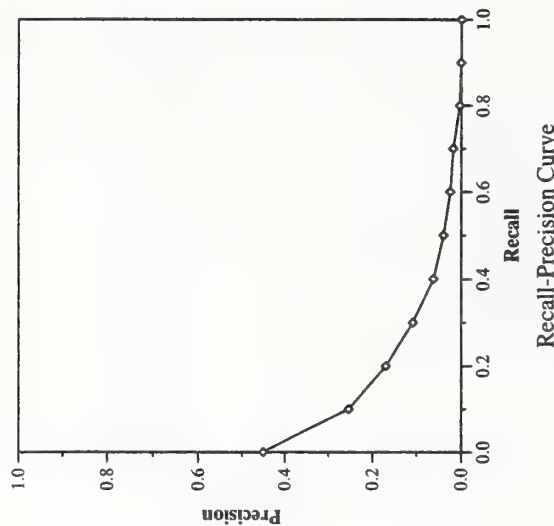


Ad hoc results — University of Surrey

Summary Statistics		
Run Number	surfah1	
Run Description	Automatic, title + desc + narr	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	1316	

Recall Level Precision Averages	
Recall	Precision
0.00	0.4508
0.10	0.2541
0.20	0.1693
0.30	0.1081
0.40	0.0617
0.50	0.0393
0.60	0.0246
0.70	0.0180
0.80	0.0036
0.90	0.0004
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0779

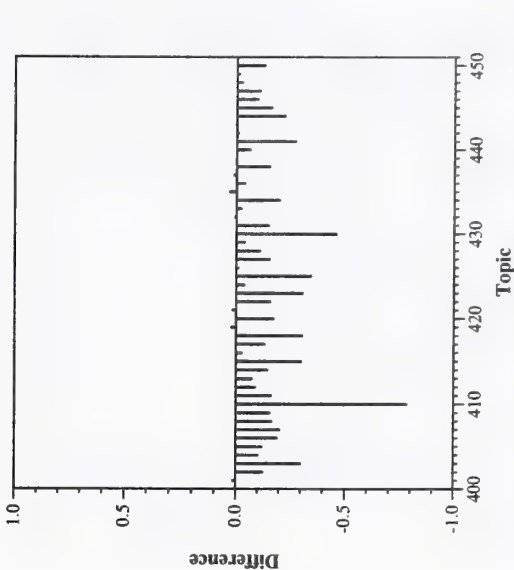
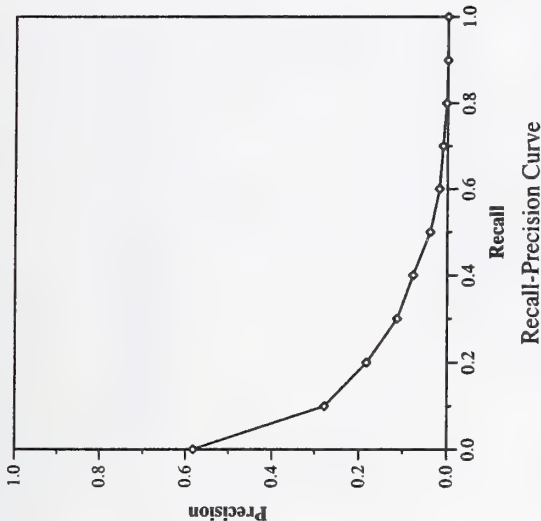
Document Level Averages	
At 5 docs	0.2280
At 10 docs	0.2020
At 15 docs	0.2040
At 20 docs	0.1960
At 30 docs	0.1720
At 100 docs	0.1190
At 200 docs	0.0837
At 500 docs	0.0435
At 1000 docs	0.0263
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1257



Summary Statistics		
Run Number	surfahi2	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	1342	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5820
0.10	0.2784
0.20	0.1827
0.30	0.1129
0.40	0.0771
0.50	0.0383
0.60	0.0180
0.70	0.0103
0.80	0.0025
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0906

Document Level Averages	
	Precision
At 5 docs	0.3360
At 10 docs	0.3020
At 15 docs	0.2680
At 20 docs	0.2390
At 30 docs	0.2027
At 100 docs	0.1080
At 200 docs	0.0775
At 500 docs	0.0442
At 1000 docs	0.0268
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1454

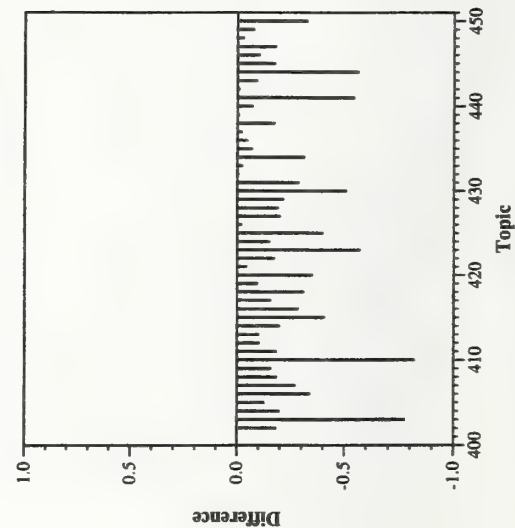
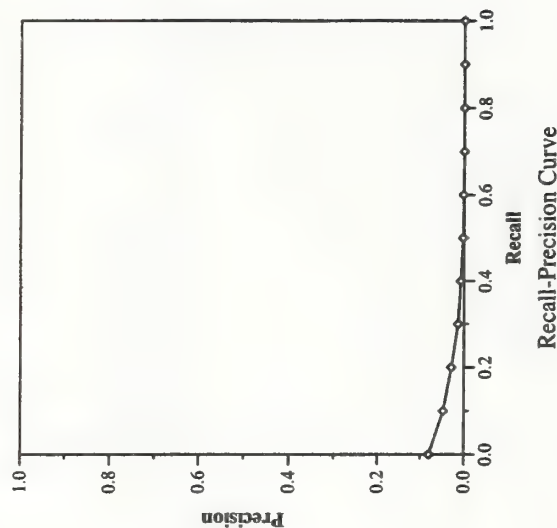


Ad hoc results — University of Surrey

Summary Statistics		
Run Number	surffal2	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	990	

Recall Level Precision Averages	
Recall	Precision
0.00	0.0801
0.10	0.0472
0.20	0.0284
0.30	0.0131
0.40	0.0071
0.50	0.0019
0.60	0.0011
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0114

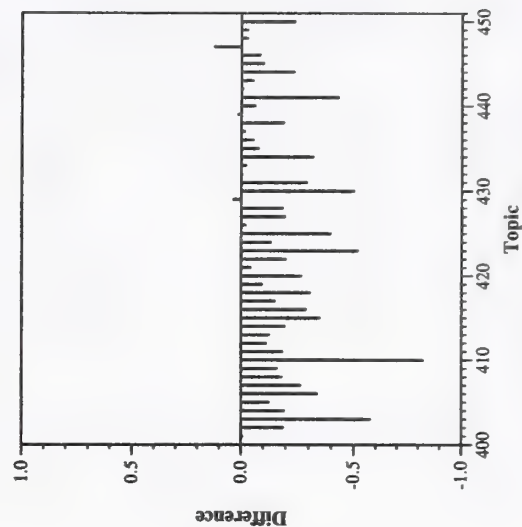
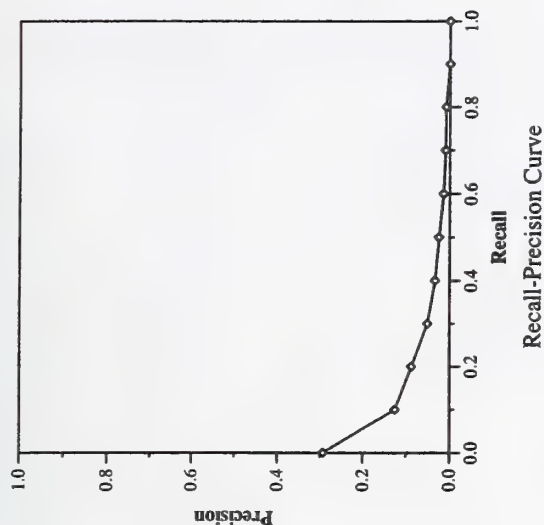
Document Level Averages	
	Precision
At 5 docs	0.0120
At 10 docs	0.0120
At 15 docs	0.0187
At 20 docs	0.0190
At 30 docs	0.0253
At 100 docs	0.0346
At 200 docs	0.0315
At 500 docs	0.0268
At 1000 docs	0.0198
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0362



Summary Statistics		
Run Number	UT810	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	776	

Recall Level Precision Averages	
Recall	Precision
0.00	0.2935
0.10	0.1253
0.20	0.0870
0.30	0.0501
0.40	0.0328
0.50	0.0240
0.60	0.0140
0.70	0.0098
0.80	0.0088
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0434

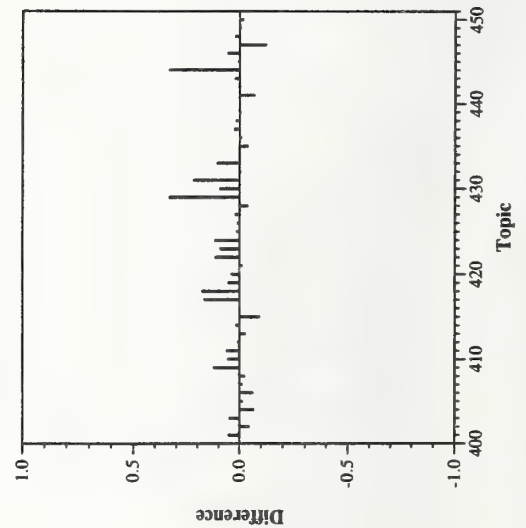
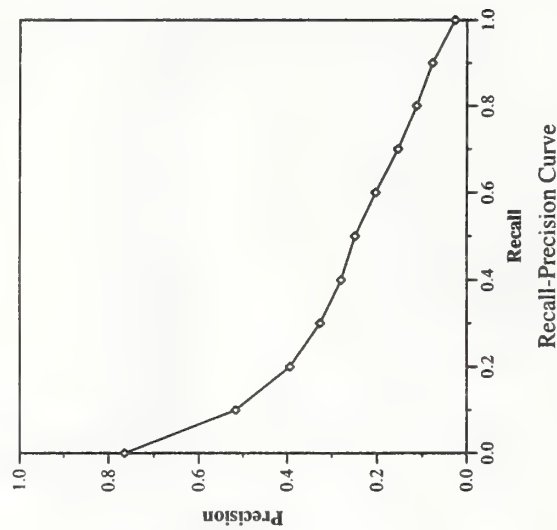
Document Level Averages	
	Precision
At 5 docs	0.1480
At 10 docs	0.1240
At 15 docs	0.1227
At 20 docs	0.1160
At 30 docs	0.0993
At 100 docs	0.0530
At 200 docs	0.0379
At 500 docs	0.0236
At 1000 docs	0.0155
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0755



Summary Statistics		
Run Number	UT800	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2908	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7652
0.10	0.5169
0.20	0.3940
0.30	0.3261
0.40	0.2792
0.50	0.2489
0.60	0.2038
0.70	0.1548
0.80	0.1136
0.90	0.0784
1.00	0.0274
Average precision over all relevant docs	
non-interpolated	0.2602

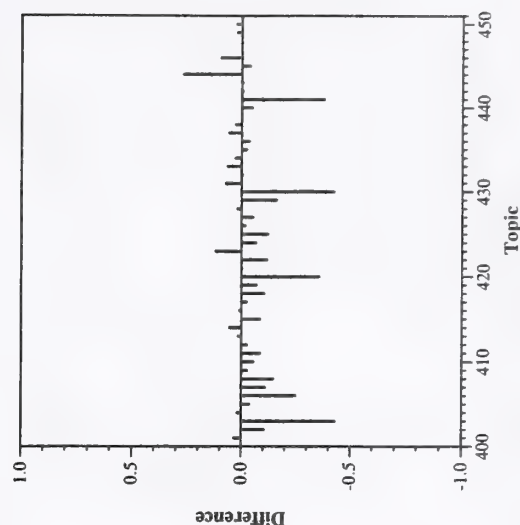
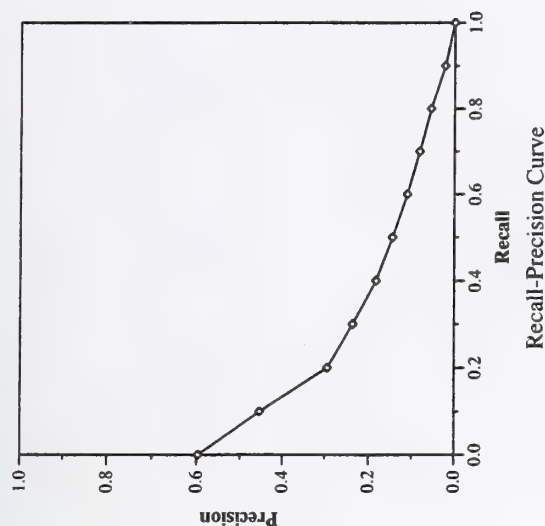
Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.5000
At 15 docs	0.4467
At 20 docs	0.4140
At 30 docs	0.3620
At 100 docs	0.2196
At 200 docs	0.1567
At 500 docs	0.0943
At 1000 docs	0.0582
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3061



Summary Statistics		
Run Number	UT803	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2424	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5968
0.10	0.4542
0.20	0.2944
0.30	0.2363
0.40	0.1831
0.50	0.1449
0.60	0.1110
0.70	0.0819
0.80	0.0553
0.90	0.0224
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1757

Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3700
At 15 docs	0.3627
At 20 docs	0.3450
At 30 docs	0.3140
At 100 docs	0.1966
At 200 docs	0.1361
At 500 docs	0.0794
At 1000 docs	0.0485
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2390

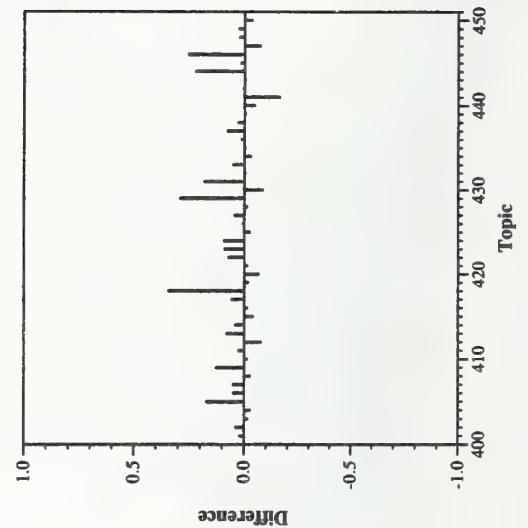
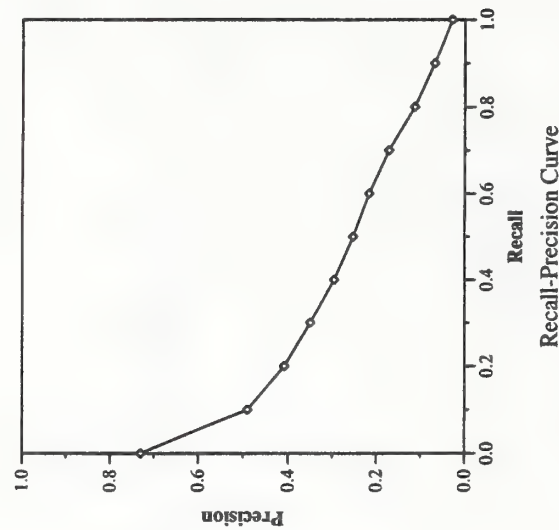


Ad hoc results — University of Twente

Summary Statistics		
Run Number	UT803b	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	3065	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7300
0.10	0.4900
0.20	0.4084
0.30	0.3501
0.40	0.2960
0.50	0.2528
0.60	0.2167
0.70	0.1719
0.80	0.1135
0.90	0.0681
1.00	0.0283
Average precision over all relevant docs	
non-interpolated	0.2598

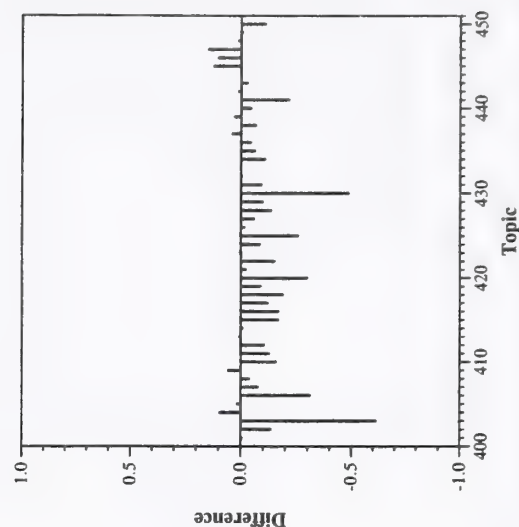
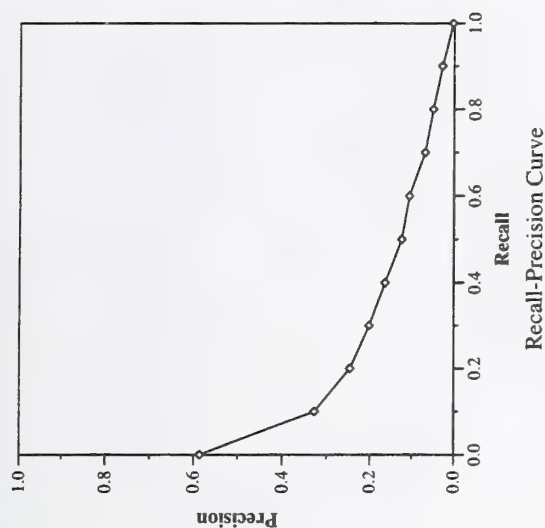
Document Level Averages	
	Precision
At 5 docs	0.4440
At 10 docs	0.4360
At 15 docs	0.4253
At 20 docs	0.3890
At 30 docs	0.3540
At 100 docs	0.2324
At 200 docs	0.1654
At 500 docs	0.0998
At 1000 docs	0.0613
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3095



Summary Statistics		
Run Number	UT813	
Run Description	Automatic, title + desc	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	4728	
Rel-ret:	2406	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5857
0.10	0.3282
0.20	0.2464
0.30	0.2018
0.40	0.1641
0.50	0.1245
0.60	0.1063
0.70	0.0689
0.80	0.0495
0.90	0.0276
1.00	0.0025
Average precision over all relevant docs	
non-interpolated	0.1449

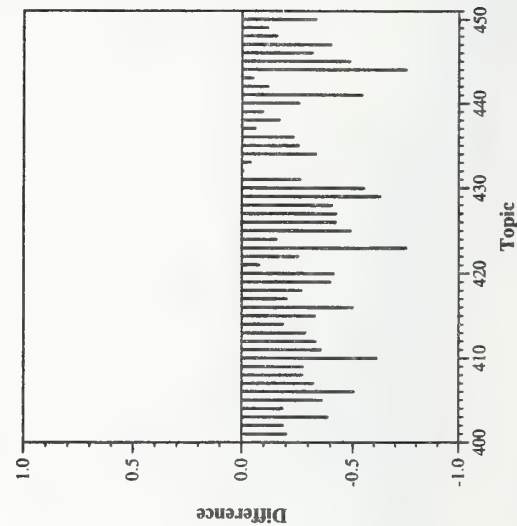
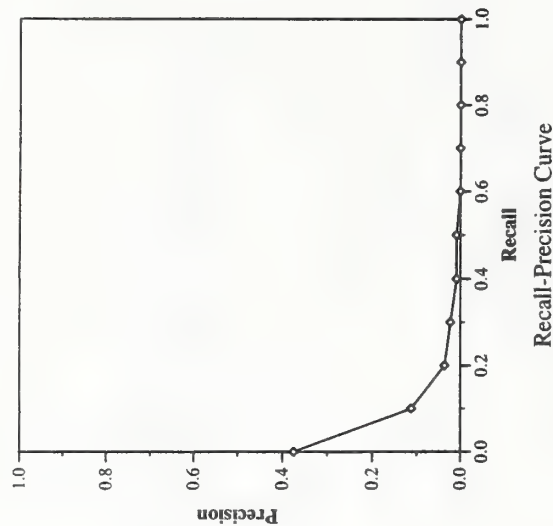
Document Level Averages	
	Precision
At 5 docs	0.3040
At 10 docs	0.2860
At 15 docs	0.2653
At 20 docs	0.2510
At 30 docs	0.2313
At 100 docs	0.1580
At 200 docs	0.1176
At 500 docs	0.0750
At 1000 docs	0.0481
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1997



Summary Statistics	
Run Number	citr82
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	592

Recall Level Precision Averages	
Recall	Precision
0.00	0.3747
0.10	0.1120
0.20	0.0359
0.30	0.0228
0.40	0.0086
0.50	0.0086
0.60	0.0003
0.70	0.0003
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0301

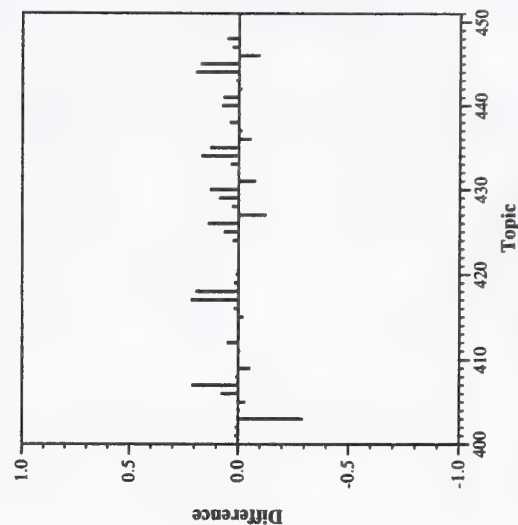
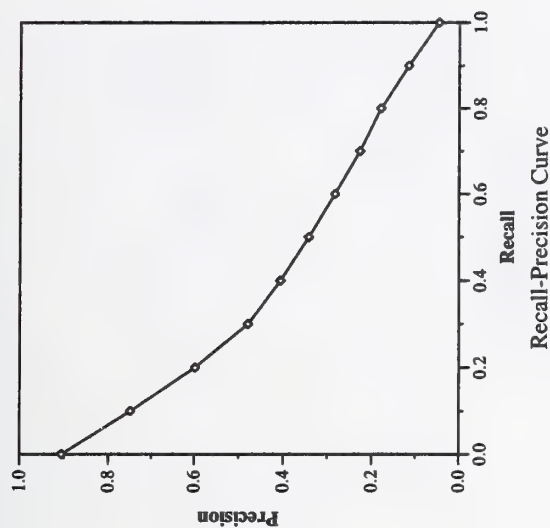
Document Level Averages	
	Precision
At 5 docs	0.2040
At 10 docs	0.1860
At 15 docs	0.1587
At 20 docs	0.1390
At 30 docs	0.1233
At 100 docs	0.0648
At 200 docs	0.0420
At 500 docs	0.0211
At 1000 docs	0.0118
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0671



Summary Statistics	
Run Number	CL99XT
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3367

Recall Level Precision Averages	
Recall	Precision
0.00	0.9061
0.10	0.7491
0.20	0.5984
0.30	0.4794
0.40	0.4061
0.50	0.3423
0.60	0.2836
0.70	0.2275
0.80	0.1797
0.90	0.1160
1.00	0.0467
Average precision over all relevant docs	
non-interpolated	0.3730

Document Level Averages	
At 5 docs	0.7680
At 10 docs	0.6920
At 15 docs	0.6280
At 20 docs	0.5730
At 30 docs	0.4907
At 100 docs	0.3078
At 200 docs	0.2148
At 500 docs	0.1173
At 1000 docs	0.0673
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3829

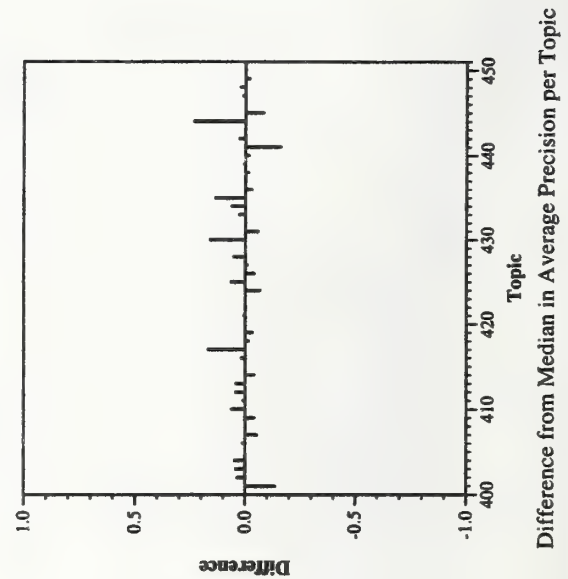
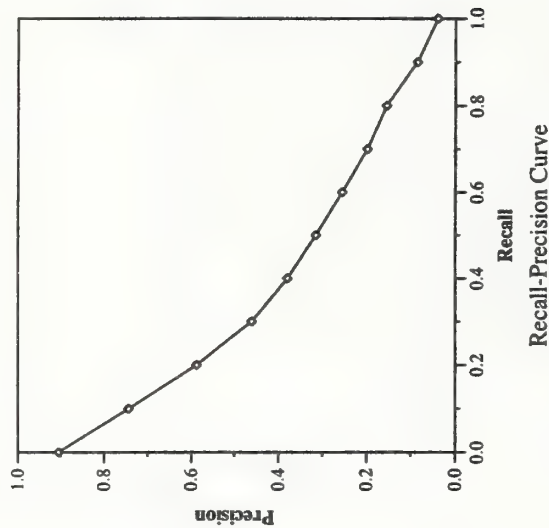


Ad hoc results — CLARITECH Corporation

Summary Statistics	
Run Number	CL99SD
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3304

Recall Level Precision Averages	
Recall	Precision
0.00	0.9047
0.10	0.7438
0.20	0.5875
0.30	0.4601
0.40	0.3794
0.50	0.3148
0.60	0.2555
0.70	0.1984
0.80	0.1549
0.90	0.0852
1.00	0.0392
Average precision over all relevant docs	
non-interpolated	0.3537

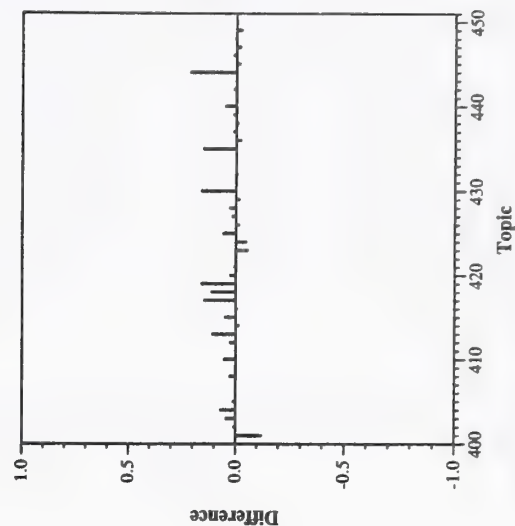
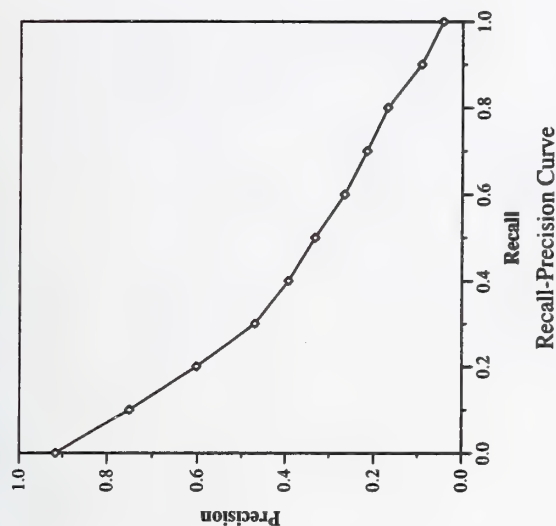
Document Level Averages	
	Precision
At 5 docs	0.7600
At 10 docs	0.7020
At 15 docs	0.6453
At 20 docs	0.5840
At 30 docs	0.4860
At 100 docs	0.2816
At 200 docs	0.1969
At 500 docs	0.1099
At 1000 docs	0.0661
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3709



Summary Statistics	
Run Number	CL99SDopt1
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3375

Recall Level Precision Averages	
Recall	Precision
0.00	0.9163
0.10	0.7509
0.20	0.6030
0.30	0.4704
0.40	0.3927
0.50	0.3320
0.60	0.2665
0.70	0.2165
0.80	0.1699
0.90	0.0924
1.00	0.0439
Average precision over all relevant docs	
non-interpolated	0.3682

Document Level Averages	
	Precision
At 5 docs	0.8000
At 10 docs	0.7080
At 15 docs	0.6440
At 20 docs	0.5870
At 30 docs	0.4887
At 100 docs	0.2862
At 200 docs	0.2046
At 500 docs	0.1132
At 1000 docs	0.0675
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3837

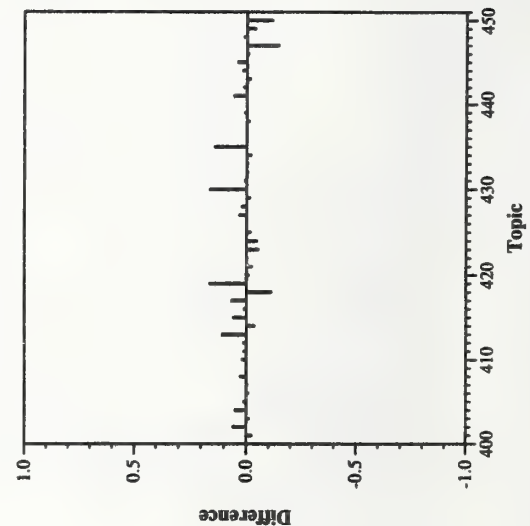
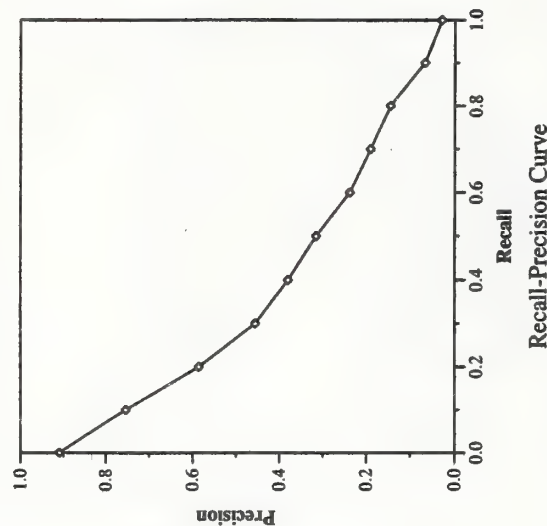


Ad hoc results — CLARITECH Corporation

Summary Statistics	
Run Number	CL99SDopt2
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3296

Recall Level Precision Averages	
Recall	Precision
0.00	0.9099
0.10	0.7540
0.20	0.5859
0.30	0.4553
0.40	0.3800
0.50	0.3154
0.60	0.2376
0.70	0.1901
0.80	0.1449
0.90	0.0666
1.00	0.0285
Average precision over all relevant docs	
non-interpolated	0.3520

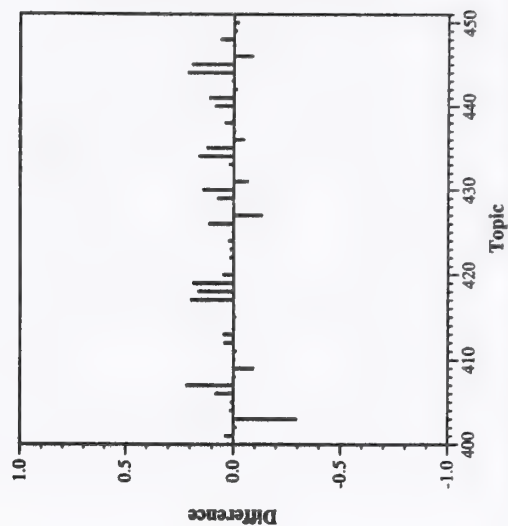
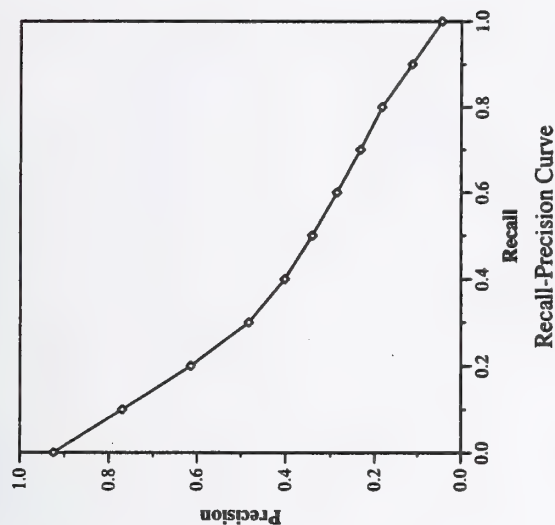
Document Level Averages	
	Precision
At 5 docs	0.7880
At 10 docs	0.7120
At 15 docs	0.6440
At 20 docs	0.5840
At 30 docs	0.4880
At 100 docs	0.2766
At 200 docs	0.1930
At 500 docs	0.1090
At 1000 docs	0.0659
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3672



Summary Statistics	
Run Number	CL99XTopt
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3366

Recall Level Precision Averages	
Recall	Precision
0.00	0.9245
0.10	0.7703
0.20	0.6148
0.30	0.4826
0.40	0.4021
0.50	0.3407
0.60	0.2862
0.70	0.2335
0.80	0.1842
0.90	0.1135
1.00	0.0443
Average precision over all relevant docs	
non-interpolated	0.3766

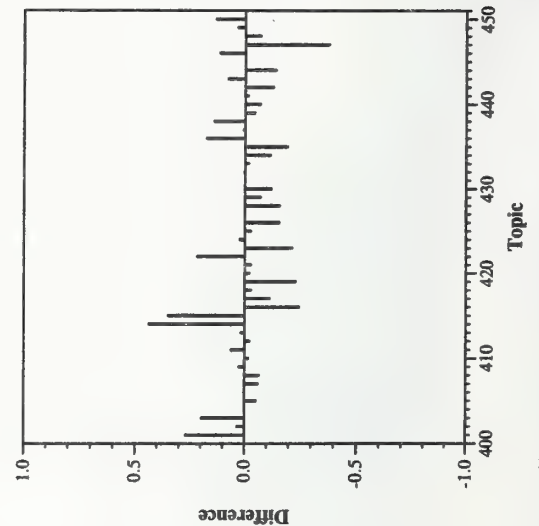
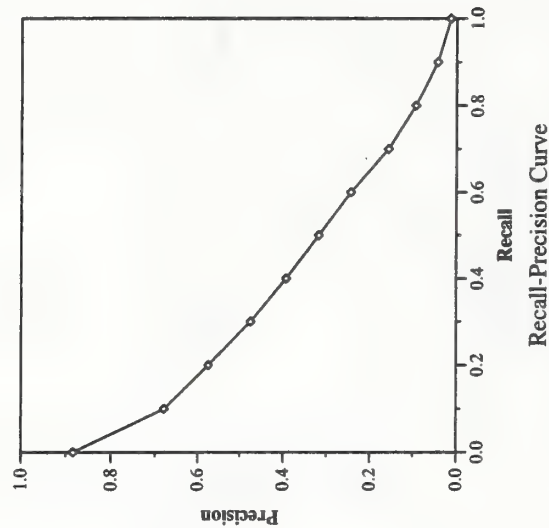
Document Level Averages	
	Precision
At 5 docs	0.7680
At 10 docs	0.6920
At 15 docs	0.6280
At 20 docs	0.5730
At 30 docs	0.4940
At 100 docs	0.3030
At 200 docs	0.2144
At 500 docs	0.1172
At 1000 docs	0.0673
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3788



Summary Statistics	
Run Number	8manexT3D1N0
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3325

Recall Level Precision Averages	
Recall	Precision
0.00	0.8844
0.10	0.6793
0.20	0.5748
0.30	0.4746
0.40	0.3933
0.50	0.3182
0.60	0.2443
0.70	0.1567
0.80	0.0935
0.90	0.0436
1.00	0.0139
Average precision over all relevant docs	
non-interpolated	0.3346

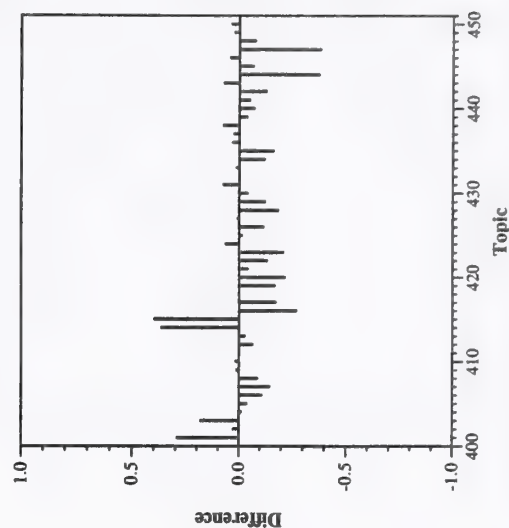
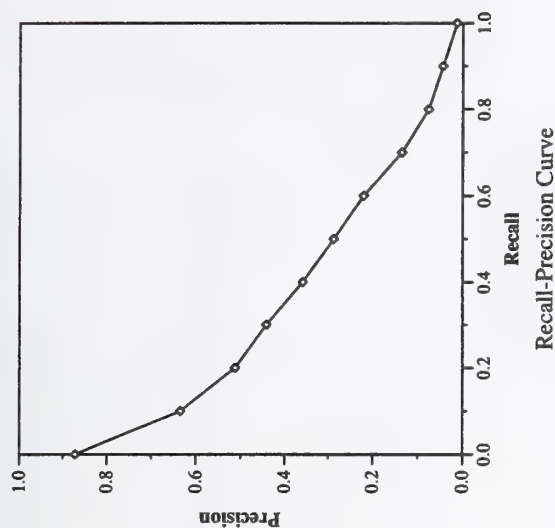
Document Level Averages	
	Precision
At 5 docs	0.7080
At 10 docs	0.6520
At 15 docs	0.5853
At 20 docs	0.5420
At 30 docs	0.4800
At 100 docs	0.2940
At 200 docs	0.2005
At 500 docs	0.1111
At 1000 docs	0.0665
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3671



Summary Statistics	
Run Number	GE8MTD2
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3264

Recall Level Precision Averages	
Recall	Precision
0.00	0.8715
0.10	0.6350
0.20	0.5126
0.30	0.4418
0.40	0.3584
0.50	0.2876
0.60	0.2202
0.70	0.1351
0.80	0.0761
0.90	0.0444
1.00	0.0133
Average precision over all relevant docs	
non-interpolated	0.3066

Document Level Averages	
	Precision
At 5 docs	0.7280
At 10 docs	0.6020
At 15 docs	0.5600
At 20 docs	0.5080
At 30 docs	0.4493
At 100 docs	0.2718
At 200 docs	0.1893
At 500 docs	0.1067
At 1000 docs	0.0653
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3422

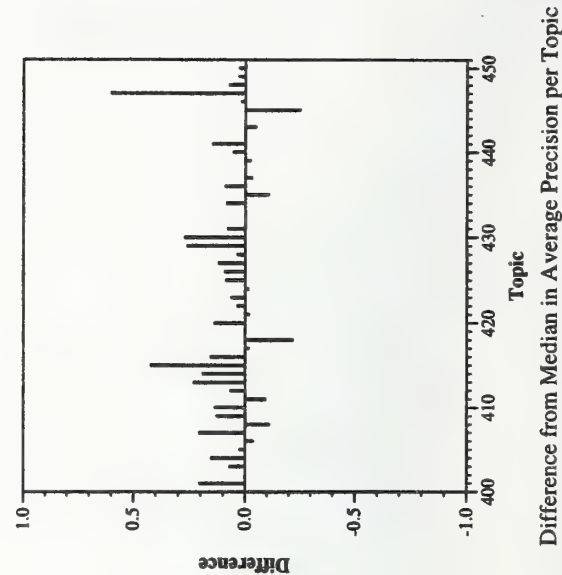
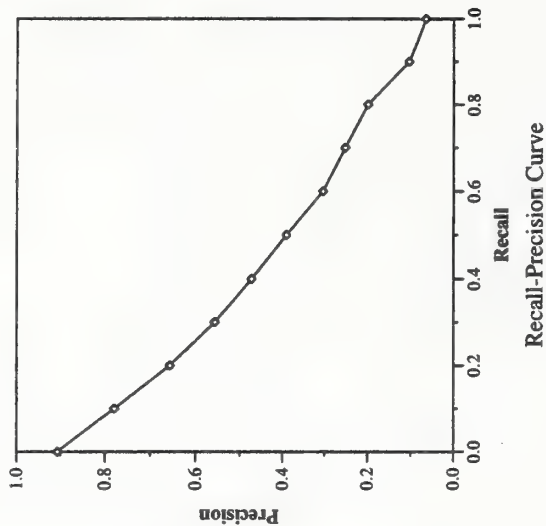


Ad hoc results --- IIT/AAT/NCR

Summary Statistics	
Run Number	iit99mal
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	32061
Relevant:	4728
Rel-ret:	3106

Recall Level Precision Averages	
Recall	Precision
0.00	0.9066
0.10	0.7790
0.20	0.6570
0.30	0.5572
0.40	0.4731
0.50	0.3908
0.60	0.3038
0.70	0.2514
0.80	0.1990
0.90	0.1042
1.00	0.0648
Average precision over all relevant docs	
non-interpolated	0.4104

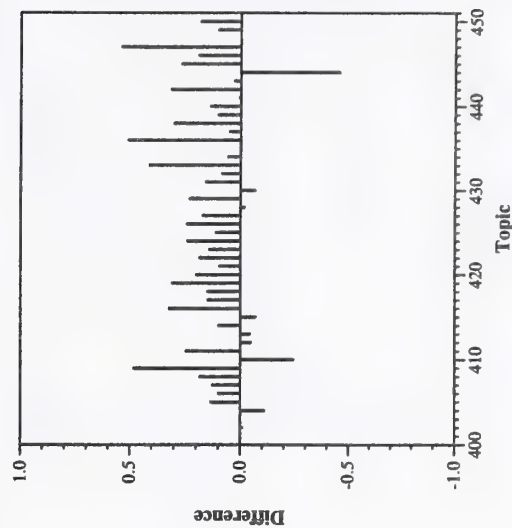
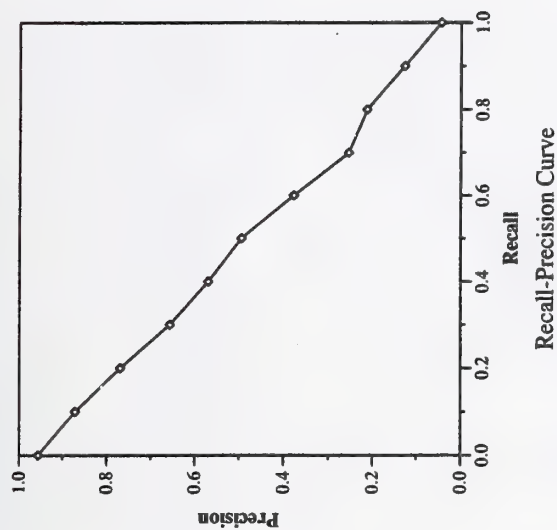
Document Level Averages	
	Precision
At 5 docs	0.7840
At 10 docs	0.7000
At 15 docs	0.6680
At 20 docs	0.6290
At 30 docs	0.5527
At 100 docs	0.3168
At 200 docs	0.2119
At 500 docs	0.1098
At 1000 docs	0.0621
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4317



Summary Statistics	
Run Number	READWARE2
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	5785
Relevant:	4728
Rel-ret:	2774

Recall Level Precision Averages	
Recall	Precision
0.00	0.9555
0.10	0.8713
0.20	0.7696
0.30	0.6570
0.40	0.5693
0.50	0.4946
0.60	0.3779
0.70	0.2552
0.80	0.2137
0.90	0.1270
1.00	0.0443
Average precision over all relevant docs	
non-interpolated	0.4692

Document Level Averages	
	Precision
At 5 docs	0.8440
At 10 docs	0.7880
At 15 docs	0.7613
At 20 docs	0.7070
At 30 docs	0.6353
At 100 docs	0.3950
At 200 docs	0.2566
At 500 docs	0.1110
At 1000 docs	0.0555
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5081

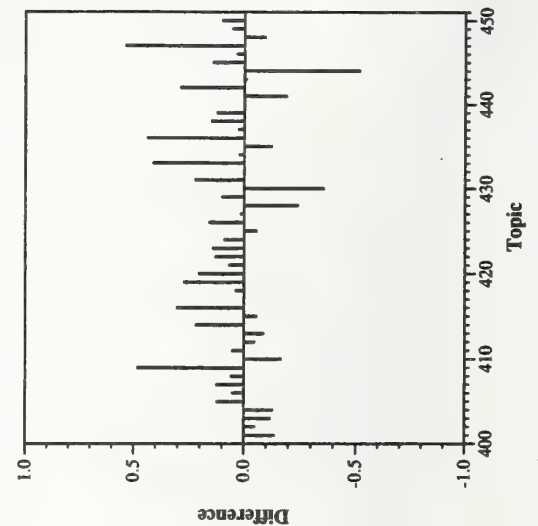
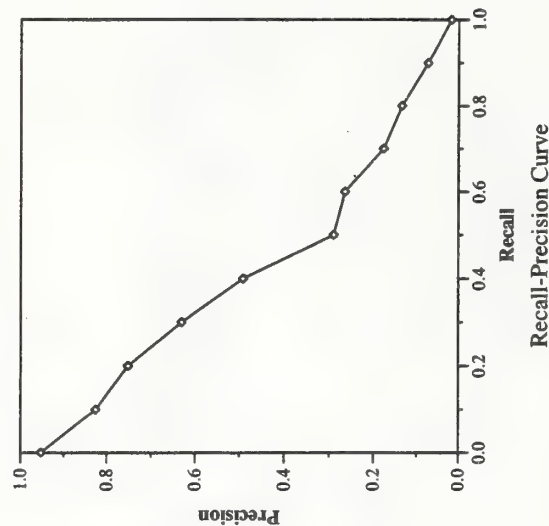


Ad hoc results — Management Information Technologies, Inc.

Summary Statistics	
Run Number	READWARE
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	3060
Relevant:	4728
Rel-ret:	2019

Recall Level Precision Averages	
Recall	Precision
0.00	0.9528
0.10	0.8255
0.20	0.7527
0.30	0.6307
0.40	0.4919
0.50	0.2905
0.60	0.2652
0.70	0.1772
0.80	0.1351
0.90	0.0731
1.00	0.0175
Average precision over all relevant docs	
non-interpolated	0.4001

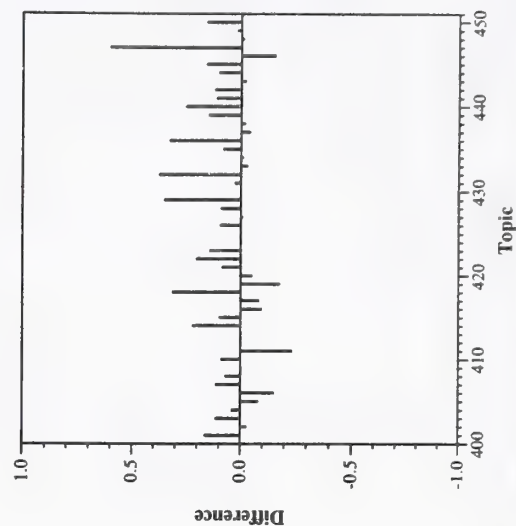
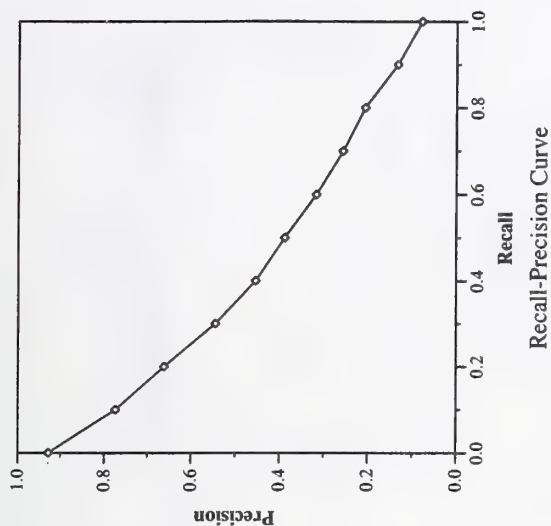
Document Level Averages	
	Precision
At 5 docs	0.8400
At 10 docs	0.7740
At 15 docs	0.7427
At 20 docs	0.6840
At 30 docs	0.6100
At 100 docs	0.3474
At 200 docs	0.2016
At 500 docs	0.0808
At 1000 docs	0.0404
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4481



Summary Statistics	
Run Number	orcl99man
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4728
Rel-ret:	3384

Recall Level Precision Averages	
Recall	Precision
0.00	0.9279
0.10	0.7747
0.20	0.6610
0.30	0.5448
0.40	0.4542
0.50	0.3882
0.60	0.3168
0.70	0.2565
0.80	0.2067
0.90	0.1332
1.00	0.0791
Average precision over all relevant docs	
non-interpolated	0.4130

Document Level Averages	
	Precision
At 5 docs	0.7840
At 10 docs	0.7220
At 15 docs	0.6587
At 20 docs	0.5990
At 30 docs	0.5367
At 100 docs	0.3440
At 200 docs	0.2282
At 500 docs	0.1206
At 1000 docs	0.0677
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4357

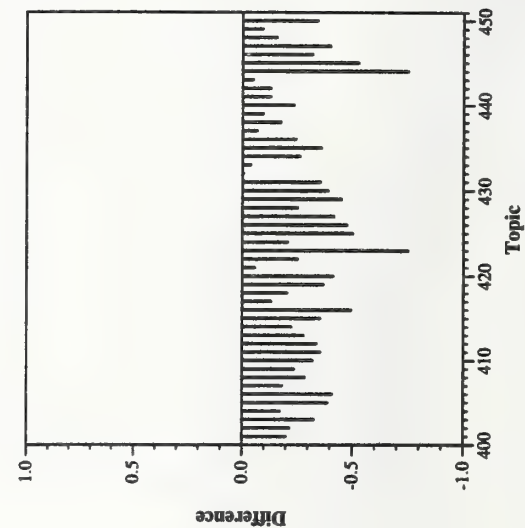
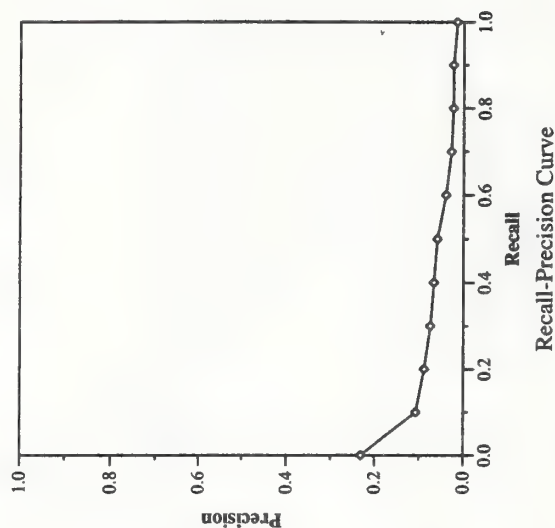


Ad hoc results — Rutgers University

Summary Statistics	
Run Number	disco1
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	37141
Relevant:	4728
Rel-ret:	1034

Recall Level Precision Averages	
Recall	Precision
0.00	0.2312
0.10	0.1072
0.20	0.0883
0.30	0.0744
0.40	0.0668
0.50	0.0592
0.60	0.0400
0.70	0.0280
0.80	0.0238
0.90	0.0233
1.00	0.0151
Average precision over all relevant docs	
non-interpolated	0.0558

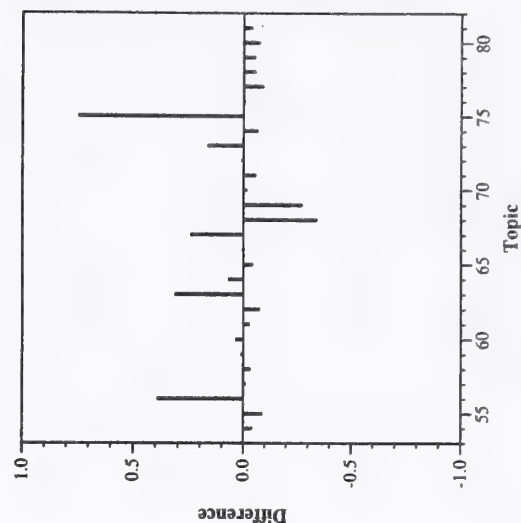
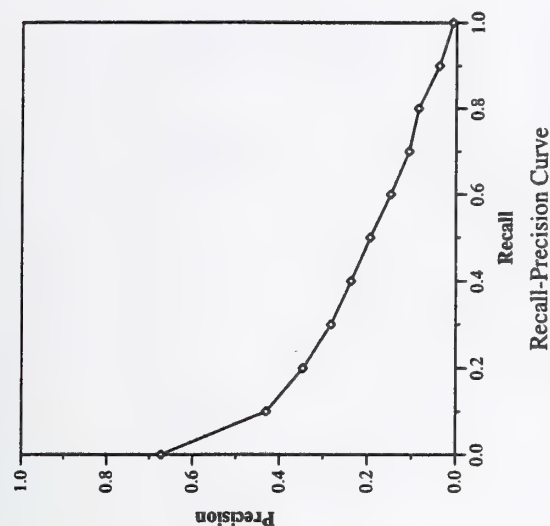
Document Level Averages	
	Precision
At 5 docs	0.1240
At 10 docs	0.0880
At 15 docs	0.1000
At 20 docs	0.0920
At 30 docs	0.0873
At 100 docs	0.0644
At 200 docs	0.0463
At 500 docs	0.0292
At 1000 docs	0.0207
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0883



Summary Statistics	
Run Number	CLARITdmwf
Run Description	english, automatic, T+D+N
Number of Topics	28
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2421
Rel-ret:	1436

Recall Level Precision Averages	
Recall	Precision
0.00	0.6726
0.10	0.4302
0.20	0.3471
0.30	0.2829
0.40	0.2368
0.50	0.1937
0.60	0.1480
0.70	0.1065
0.80	0.0855
0.90	0.0381
1.00	0.0079
Average precision over all relevant docs	
non-interpolated	0.2127

Document Level Averages	
	Precision
At 5 docs	0.5000
At 10 docs	0.4536
At 15 docs	0.3952
At 20 docs	0.3661
At 30 docs	0.3286
At 100 docs	0.2257
At 200 docs	0.1564
At 500 docs	0.0874
At 1000 docs	0.0513
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2473

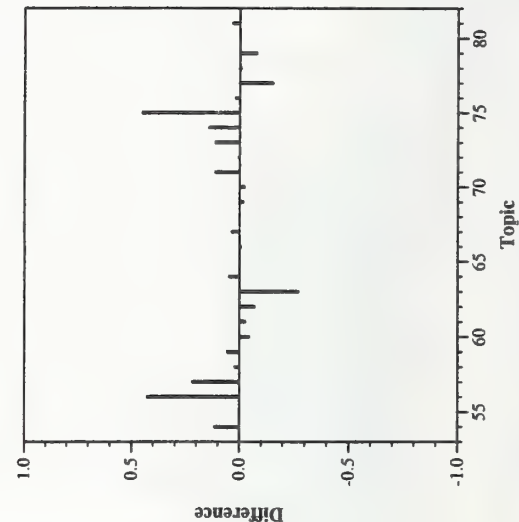
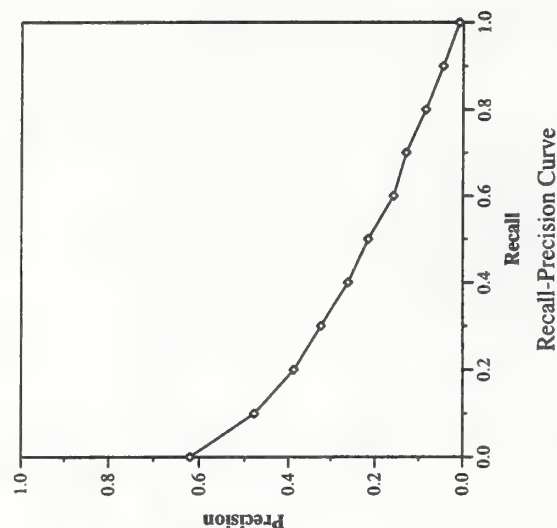


Cross-language track results — CLARITECH Corporation

Summary Statistics	
Run Number	CLARITrmwfl
Run Description	english, automatic, T+D+N
Number of Topics	28
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2421
Rel-ret:	1807

Recall Level Precision Averages	
Recall	Precision
0.00	0.6198
0.10	0.4785
0.20	0.3877
0.30	0.3236
0.40	0.2610
0.50	0.2153
0.60	0.1584
0.70	0.1298
0.80	0.0851
0.90	0.0453
1.00	0.0086
Average precision over all relevant docs	
non-interpolated	0.2297

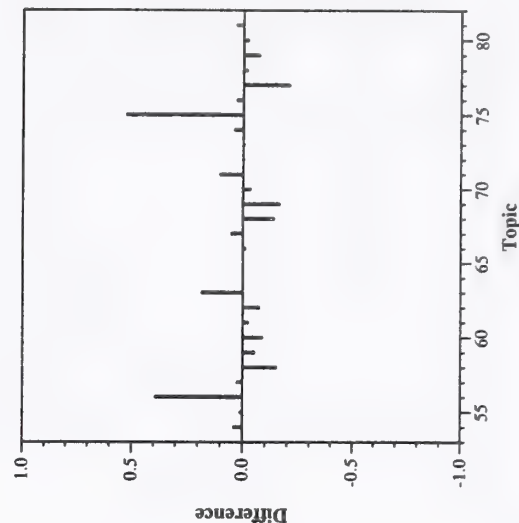
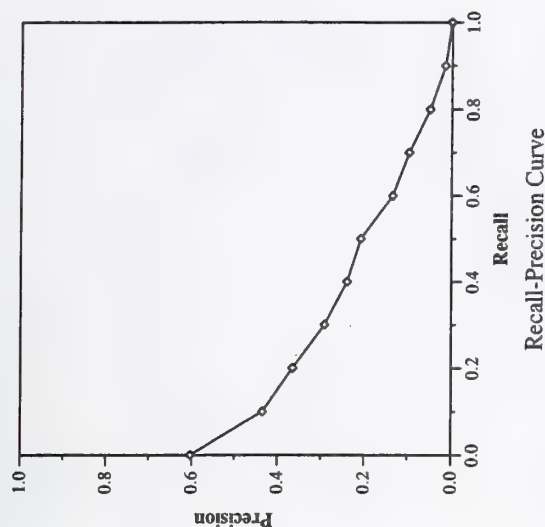
Document Level Averages	
	Precision
At 5 docs	0.4643
At 10 docs	0.4321
At 15 docs	0.4071
At 20 docs	0.3875
At 30 docs	0.3571
At 100 docs	0.2661
At 200 docs	0.1918
At 500 docs	0.1097
At 1000 docs	0.0645
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2717



Summary Statistics	
Run Number	CLARITrmwf2
Run Description	english, automatic, T+D+N
Number of Topics	28
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2421
Rel-ret:	1626

Recall Level Precision Averages	
Recall	Precision
0.00	0.6032
0.10	0.4348
0.20	0.3653
0.30	0.2919
0.40	0.2400
0.50	0.2081
0.60	0.1347
0.70	0.0973
0.80	0.0495
0.90	0.0152
1.00	0.0009
Average precision over all relevant docs	
non-interpolated	0.2036

Document Level Averages	
	Precision
At 5 docs	0.4214
At 10 docs	0.4179
At 15 docs	0.3881
At 20 docs	0.3643
At 30 docs	0.3393
At 100 docs	0.2429
At 200 docs	0.1739
At 500 docs	0.0967
At 1000 docs	0.0581
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2514

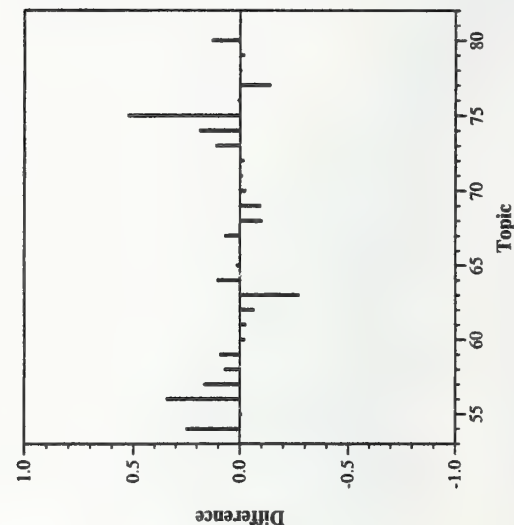
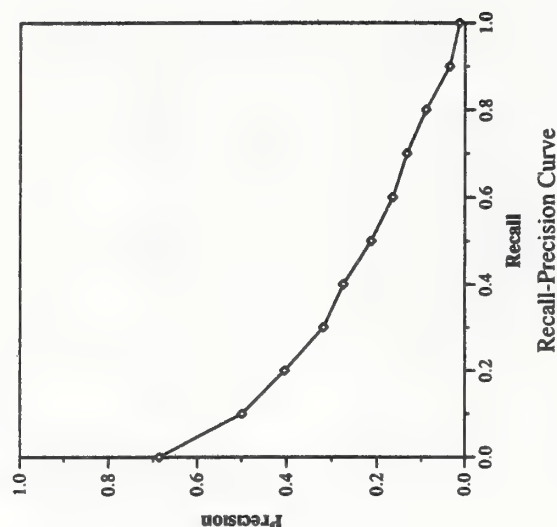


Cross-language track results — CLARITECH Corporation

Summary Statistics	
Run Number	CLARITrmwf3
Run Description	english, automatic, T+D+N
Number of Topics	28
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2421
Rel-ret:	1789

Recall Level Precision Averages	
Recall	Precision
0.00	0.6865
0.10	0.4991
0.20	0.4039
0.30	0.3180
0.40	0.2744
0.50	0.2133
0.60	0.1655
0.70	0.1336
0.80	0.0901
0.90	0.0349
1.00	0.0128
Average precision over all relevant docs	
non-interpolated	0.2357

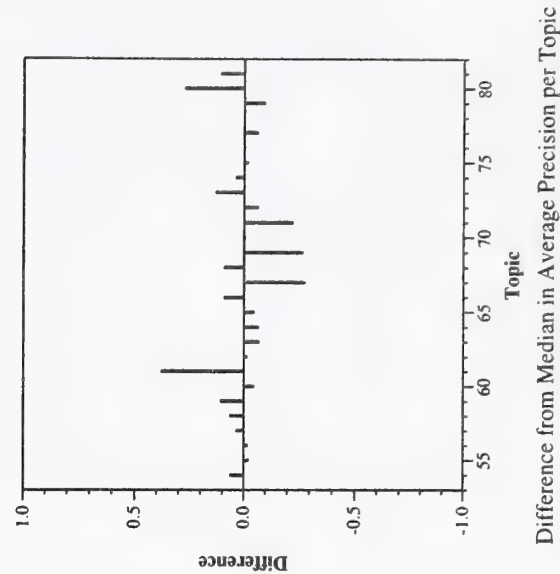
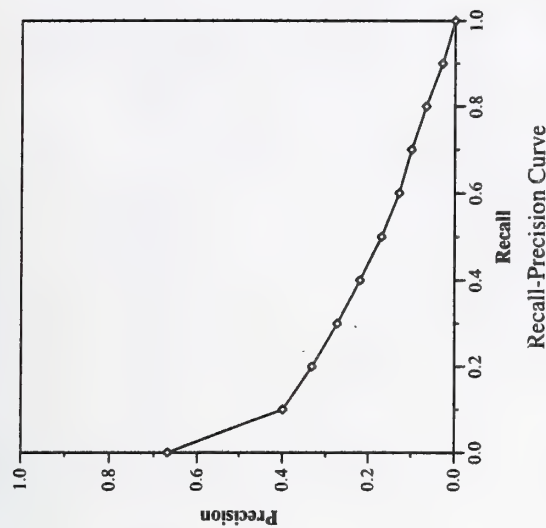
Document Level Averages	
	Precision
At 5 docs	0.4571
At 10 docs	0.4393
At 15 docs	0.4310
At 20 docs	0.4143
At 30 docs	0.3798
At 100 docs	0.2475
At 200 docs	0.1904
At 500 docs	0.1067
At 1000 docs	0.0639
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2809



Summary Statistics	
Run Number	EIT99mta
Run Description	german, automatic, T+D+N
Number of Topics	28
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2421
Rel-ret:	1639

Recall Level Precision Averages	
Recall	Precision
0.00	0.6689
0.10	0.3989
0.20	0.3322
0.30	0.2730
0.40	0.2201
0.50	0.1695
0.60	0.1283
0.70	0.0997
0.80	0.0657
0.90	0.0285
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1937

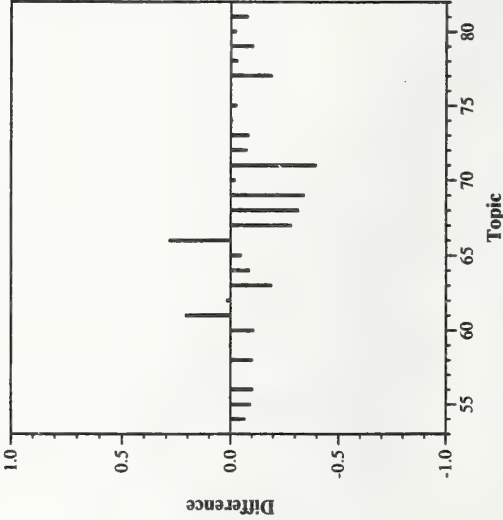
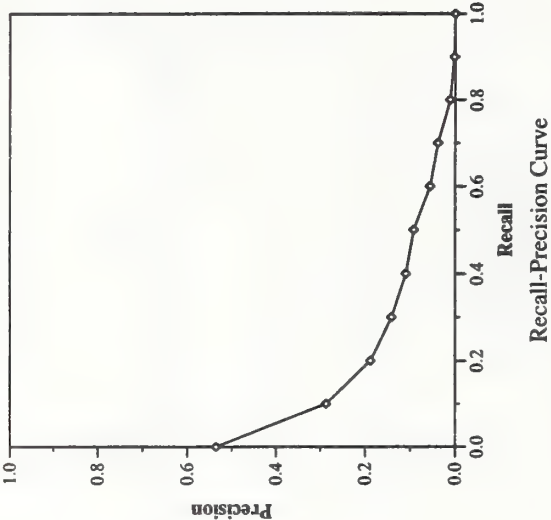
Document Level Averages	
	Precision
At 5 docs	0.4500
At 10 docs	0.3786
At 15 docs	0.3429
At 20 docs	0.3321
At 30 docs	0.3095
At 100 docs	0.2168
At 200 docs	0.1693
At 500 docs	0.0975
At 1000 docs	0.0585
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2415



Summary Statistics		
Run Number	EIT99sal	
Run Description	german, automatic, T+D	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1100	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5354
0.10	0.2868
0.20	0.1881
0.30	0.1410
0.40	0.1093
0.50	0.0920
0.60	0.0557
0.70	0.0380
0.80	0.0114
0.90	0.0022
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1108

Document Level Averages	
	Precision
At 5 docs	0.3143
At 10 docs	0.2750
At 15 docs	0.2595
At 20 docs	0.2286
At 30 docs	0.2119
At 100 docs	0.1529
At 200 docs	0.1079
At 500 docs	0.0620
At 1000 docs	0.0393
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1682

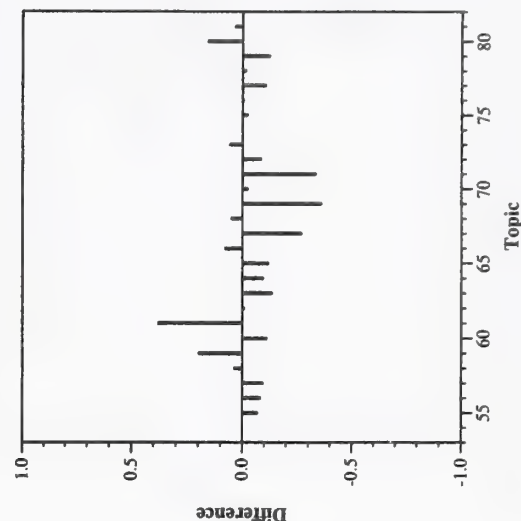
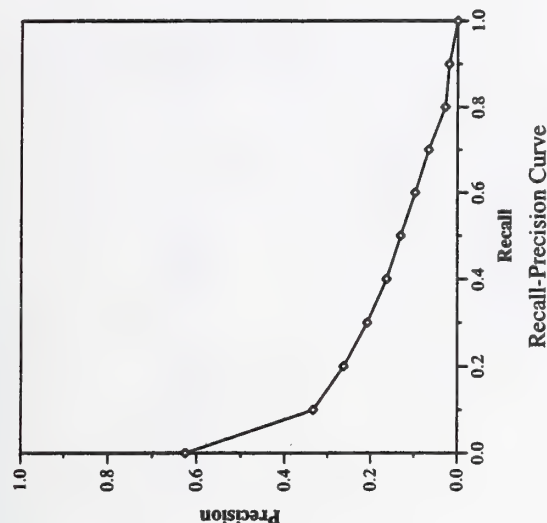


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	EIT99sta
Run Description	german, automatic, T+D+N
Number of Topics	28
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2421
Ret-ret:	1254

Recall Level Precision Averages	
Recall	Precision
0.00	0.6251
0.10	0.3339
0.20	0.2631
0.30	0.2080
0.40	0.1632
0.50	0.1307
0.60	0.0978
0.70	0.0666
0.80	0.0284
0.90	0.0197
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1527

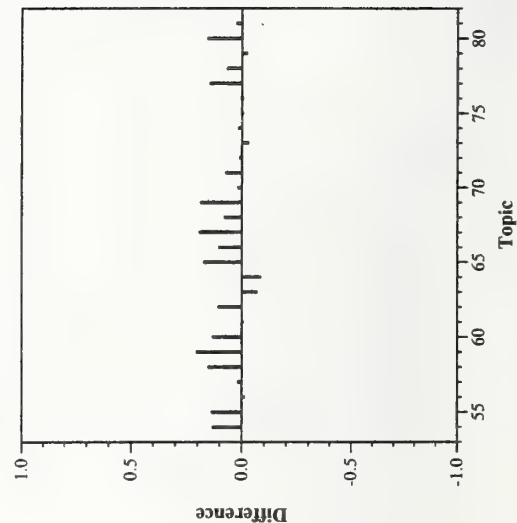
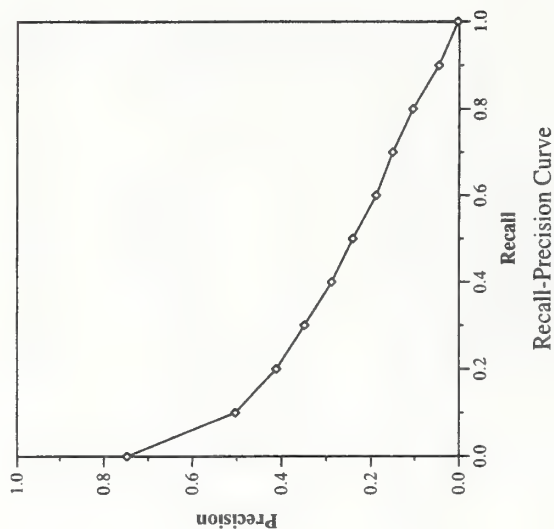
Document Level Averages	
	Precision
At 5 docs	0.3857
At 10 docs	0.3536
At 15 docs	0.3238
At 20 docs	0.2946
At 30 docs	0.2714
At 100 docs	0.1832
At 200 docs	0.1268
At 500 docs	0.0723
At 1000 docs	0.0448
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2006



Summary Statistics	
Run Number	ibmcl8ea
Run Description	english, automatic, T+D+N
Number of Topics	28
Total number of documents over all topics	
Retrieved:	27999
Relevant:	2421
Rel-ret:	1848

Recall Level Precision Averages	
Recall	Precision
0.00	0.7479
0.10	0.5033
0.20	0.4114
0.30	0.3482
0.40	0.2878
0.50	0.2411
0.60	0.1885
0.70	0.1509
0.80	0.1047
0.90	0.0458
1.00	0.0032
Average precision over all relevant docs	
non-interpolated	0.2569

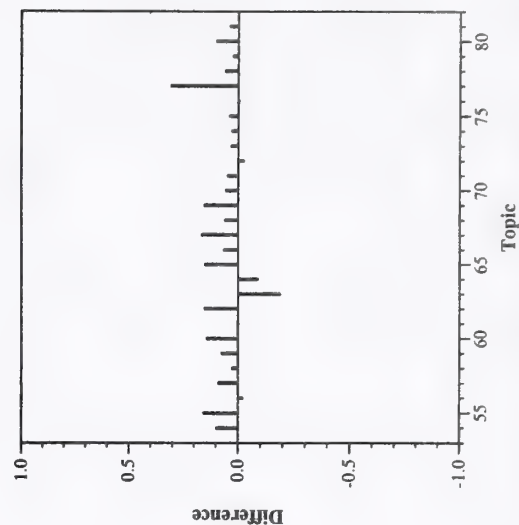
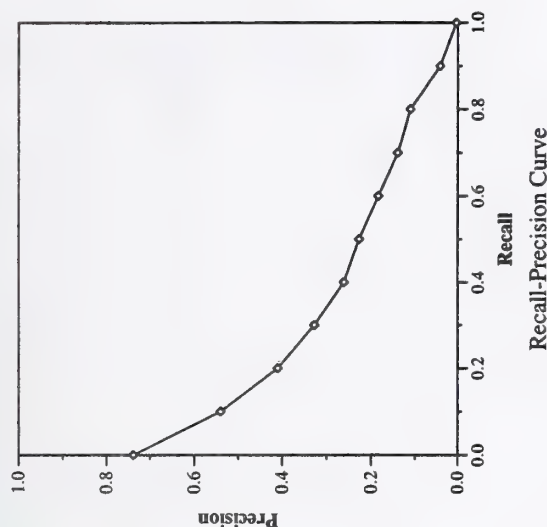
Document Level Averages	
	Precision
At 5 docs	0.5214
At 10 docs	0.5000
At 15 docs	0.4833
At 20 docs	0.4482
At 30 docs	0.4083
At 100 docs	0.2679
At 200 docs	0.1996
At 500 docs	0.1121
At 1000 docs	0.0660
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2974



Summary Statistics		
Run Number	ibmcl8ec	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1829	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7381
0.10	0.5402
0.20	0.4099
0.30	0.3275
0.40	0.2615
0.50	0.2274
0.60	0.1838
0.70	0.1396
0.80	0.1110
0.90	0.0413
1.00	0.0036
Average precision over all relevant docs	
non-interpolated	0.2515

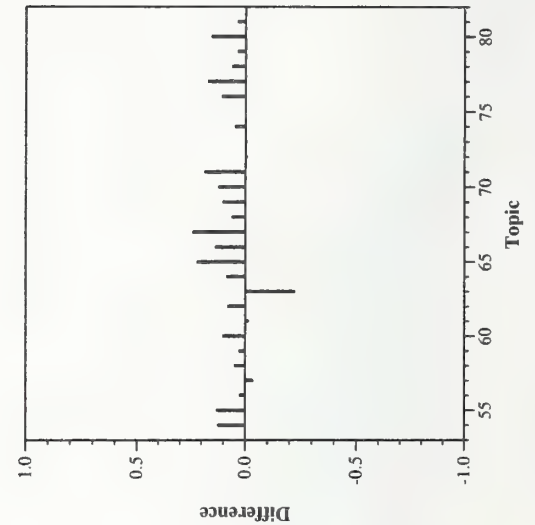
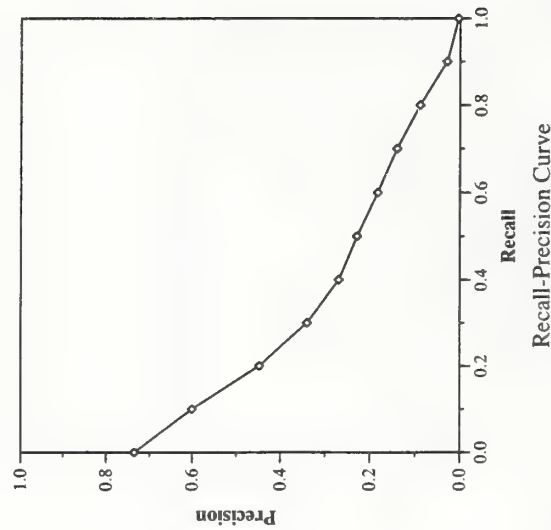
Document Level Averages	
	Precision
At 5 docs	0.5143
At 10 docs	0.4750
At 15 docs	0.4524
At 20 docs	0.4357
At 30 docs	0.3976
At 100 docs	0.2689
At 200 docs	0.1939
At 500 docs	0.1136
At 1000 docs	0.0653
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2863



Summary Statistics	
Run Number	ibmcl8fa
Run Description	french, automatic, T+D+N
Number of Topics	28
Total number of documents over all topics	
Retrieved:	27999
Relevant:	2421
Rel-ret:	1844

Recall Level Precision Averages	
Recall	Precision
0.00	0.7351
0.10	0.6004
0.20	0.4481
0.30	0.3399
0.40	0.2692
0.50	0.2284
0.60	0.1829
0.70	0.1401
0.80	0.0894
0.90	0.0292
1.00	0.0044
Average precision over all relevant docs	
non-interpolated	0.2613

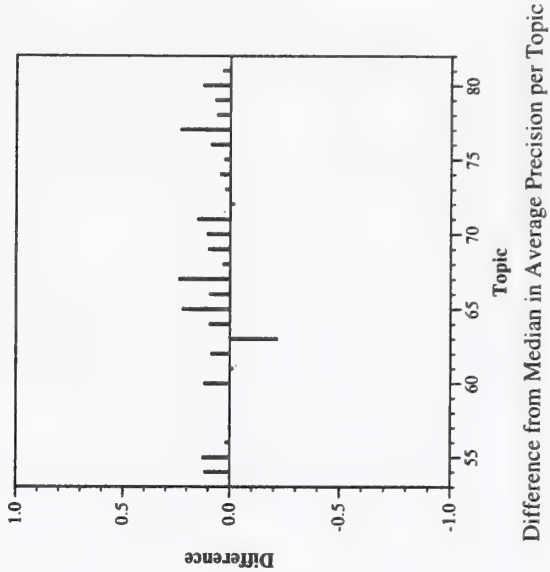
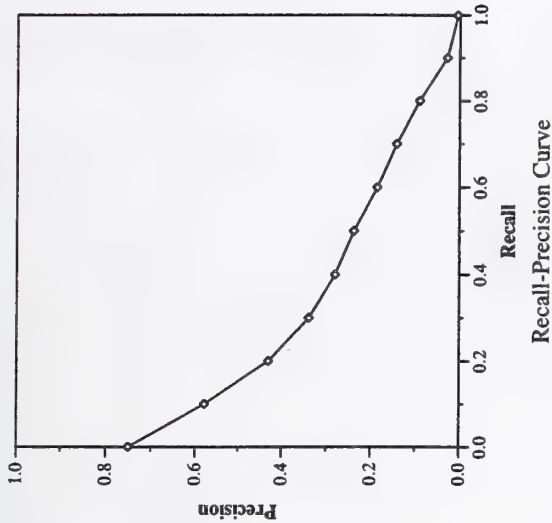
Document Level Averages	
	Precision
At 5 docs	0.5786
At 10 docs	0.5143
At 15 docs	0.4929
At 20 docs	0.4679
At 30 docs	0.4298
At 100 docs	0.2786
At 200 docs	0.2030
At 500 docs	0.1128
At 1000 docs	0.0659
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3016



Summary Statistics		
Run Number	ibmcl8fc	
Run Description	french, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1845	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7504
0.10	0.5789
0.20	0.4300
0.30	0.3387
0.40	0.2798
0.50	0.2373
0.60	0.1849
0.70	0.1412
0.80	0.0904
0.90	0.0281
1.00	0.0048
Average precision over all relevant docs	
non-interpolated	0.2600

Document Level Averages	
	Precision
At 5 docs	0.5357
At 10 docs	0.5036
At 15 docs	0.4762
At 20 docs	0.4482
At 30 docs	0.4083
At 100 docs	0.2832
At 200 docs	0.2034
At 500 docs	0.1123
At 1000 docs	0.0659
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3025

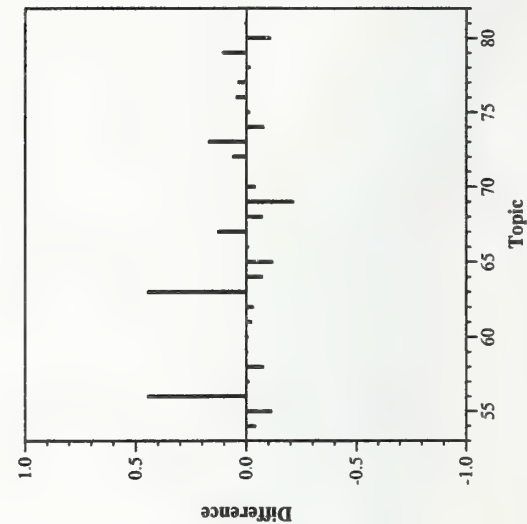
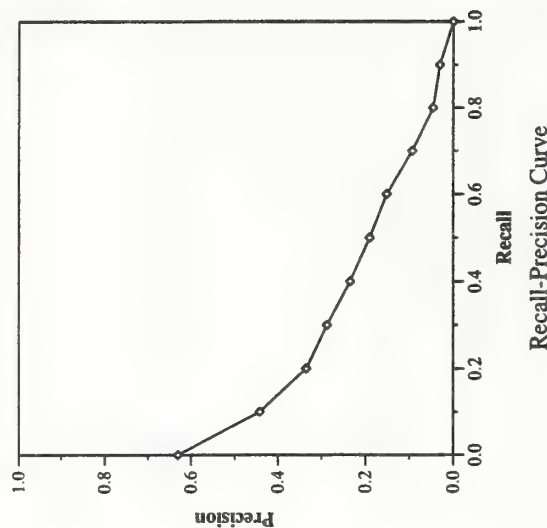


Cross-language track results — IIRIT/SIG

Summary Statistics	
Run Number	Mer8Can2x
Run Description	english, automatic, T+D+N
Number of Topics	28
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2421
Rel-ret:	1420

Recall Level Precision Averages	
Recall	Precision
0.00	0.6300
0.10	0.4421
0.20	0.3350
0.30	0.2875
0.40	0.2348
0.50	0.1901
0.60	0.1514
0.70	0.0933
0.80	0.0456
0.90	0.0303
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.2051

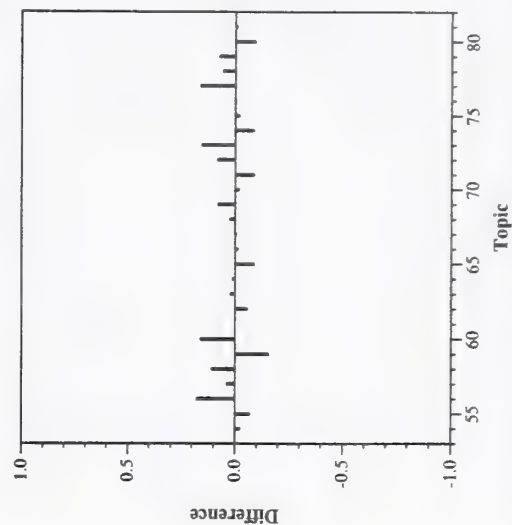
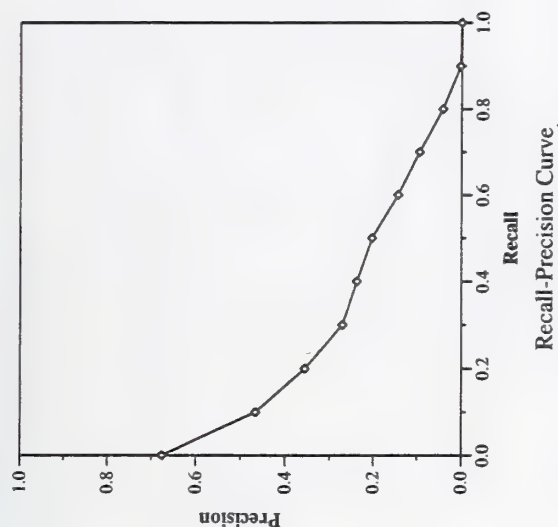
Document Level Averages	
	Precision
At 5 docs	0.5000
At 10 docs	0.4107
At 15 docs	0.3857
At 20 docs	0.3643
At 30 docs	0.3393
At 100 docs	0.2518
At 200 docs	0.1688
At 500 docs	0.0886
At 1000 docs	0.0507
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2386



Summary Statistics		
Run Number	Mer8Can2x0	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1482	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6761
0.10	0.4667
0.20	0.3564
0.30	0.2710
0.40	0.2383
0.50	0.2034
0.60	0.1446
0.70	0.0956
0.80	0.0425
0.90	0.0026
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.2059

Document Level Averages	
	Precision
At 5 docs	0.4786
At 10 docs	0.4607
At 15 docs	0.4095
At 20 docs	0.3804
At 30 docs	0.3417
At 100 docs	0.2368
At 200 docs	0.1723
At 500 docs	0.0937
At 1000 docs	0.0529
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2529

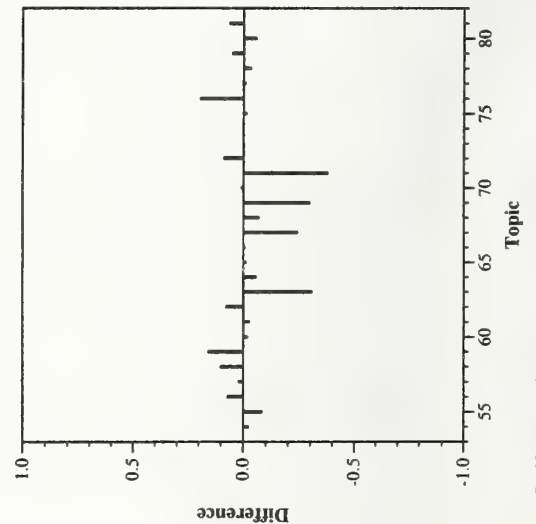
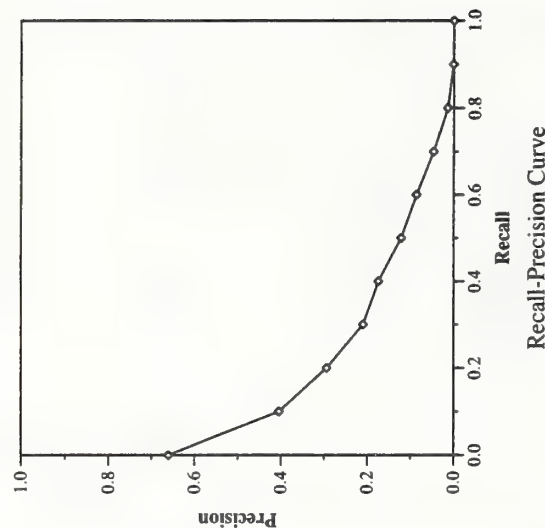


Cross-language track results — IRT/SIG

Summary Statistics		
Run Number	Mer8Cfr2x	
Run Description	french, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1342	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6604
0.10	0.4042
0.20	0.2939
0.30	0.2095
0.40	0.1737
0.50	0.1208
0.60	0.0854
0.70	0.0464
0.80	0.0143
0.90	0.0009
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1620

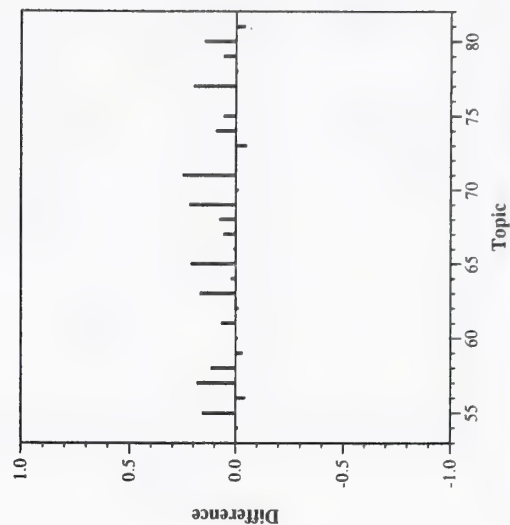
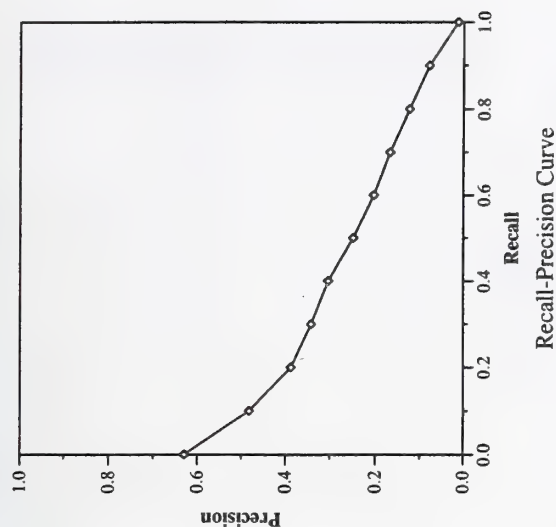
Document Level Averages	
	Precision
At 5 docs	0.4857
At 10 docs	0.4179
At 15 docs	0.3762
At 20 docs	0.3464
At 30 docs	0.3036
At 100 docs	0.1896
At 200 docs	0.1377
At 500 docs	0.0782
At 1000 docs	0.0479
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2031



Summary Statistics		
Run Number	apxl1	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1911	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6299
0.10	0.4824
0.20	0.3876
0.30	0.3421
0.40	0.3039
0.50	0.2488
0.60	0.2038
0.70	0.1675
0.80	0.1241
0.90	0.0789
1.00	0.0127
Average precision over all relevant docs	
non-interpolated	0.2571

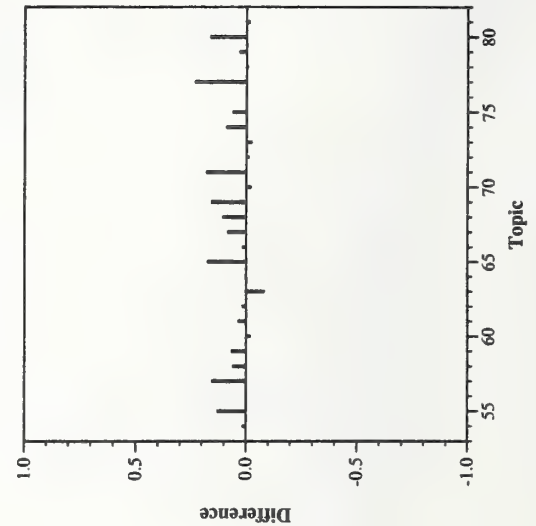
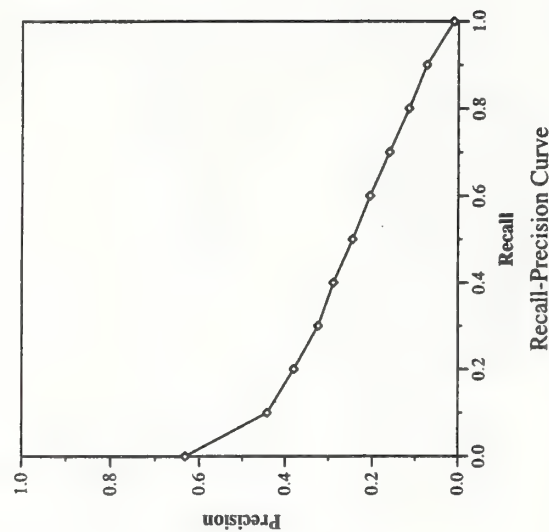
Document Level Averages	
	Precision
At 5 docs	0.4571
At 10 docs	0.4321
At 15 docs	0.4071
At 20 docs	0.3982
At 30 docs	0.3881
At 100 docs	0.2714
At 200 docs	0.2045
At 500 docs	0.1154
At 1000 docs	0.0683
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2789



Summary Statistics		
Run Number	aplx12	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1944	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6324
0.10	0.4419
0.20	0.3801
0.30	0.3243
0.40	0.2891
0.50	0.2456
0.60	0.2062
0.70	0.1614
0.80	0.1160
0.90	0.0740
1.00	0.0102
Average precision over all relevant docs	
non-interpolated	0.2471

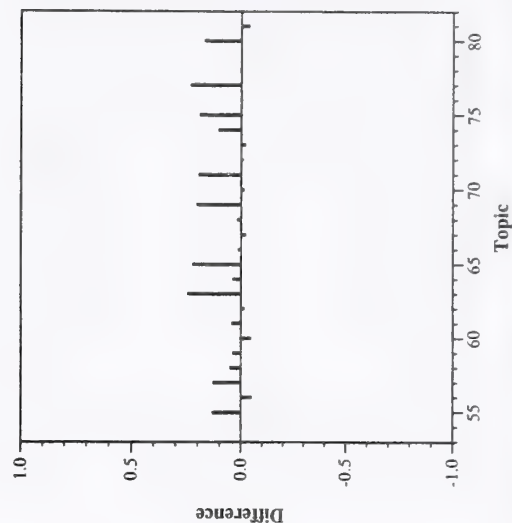
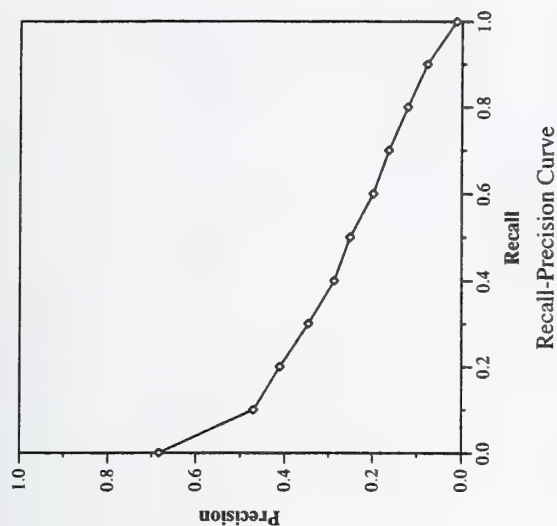
Document Level Averages	
	Precision
At 5 docs	0.4286
At 10 docs	0.4071
At 15 docs	0.3857
At 20 docs	0.3839
At 30 docs	0.3536
At 100 docs	0.2782
At 200 docs	0.2048
At 500 docs	0.1188
At 1000 docs	0.0694
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2913



Summary Statistics		
Run Number	apxl3	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1890	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6831
0.10	0.4701
0.20	0.4110
0.30	0.3461
0.40	0.2876
0.50	0.2517
0.60	0.1992
0.70	0.1640
0.80	0.1212
0.90	0.0779
1.00	0.0123
Average precision over all relevant docs	
non-interpolated	0.2542

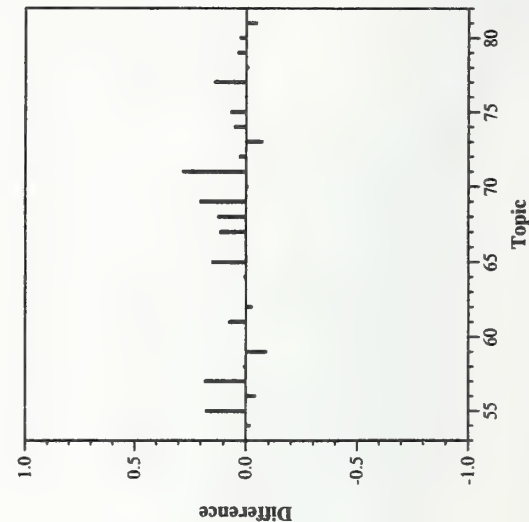
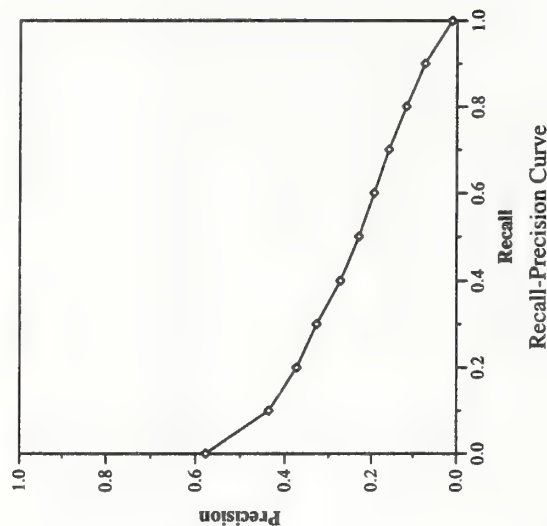
Document Level Averages	
	Precision
At 5 docs	0.4357
At 10 docs	0.4179
At 15 docs	0.4286
At 20 docs	0.4143
At 30 docs	0.3845
At 100 docs	0.2711
At 200 docs	0.1979
At 500 docs	0.1129
At 1000 docs	0.0675
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2922



Summary Statistics		
Run Number	apxl4	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1902	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5770
0.10	0.4360
0.20	0.3734
0.30	0.3279
0.40	0.2731
0.50	0.2298
0.60	0.1939
0.70	0.1584
0.80	0.1168
0.90	0.0734
1.00	0.0112
Average precision over all relevant docs	
non-interpolated	0.2389

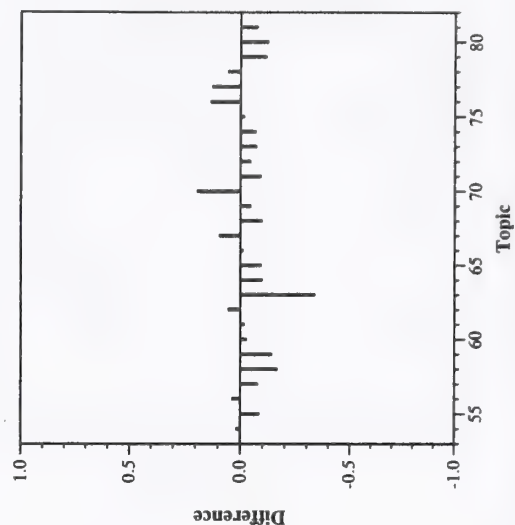
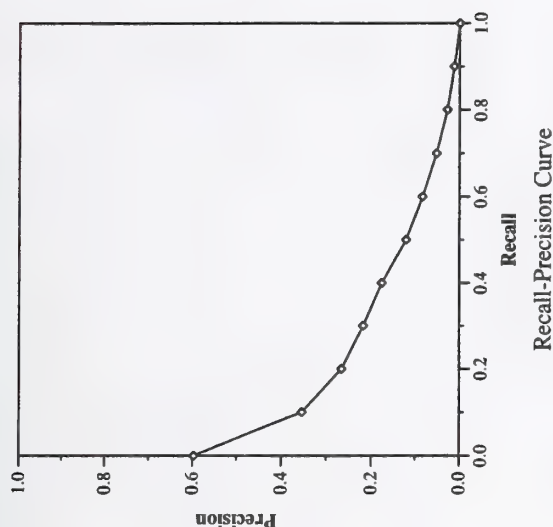
Document Level Averages	
	Precision
At 5 docs	0.3929
At 10 docs	0.3821
At 15 docs	0.3833
At 20 docs	0.3643
At 30 docs	0.3452
At 100 docs	0.2575
At 200 docs	0.2004
At 500 docs	0.1149
At 1000 docs	0.0679
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2612



Summary Statistics		
Run Number	nmsuil	
Run Description	english, manual, T+D	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1420	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5969
0.10	0.3540
0.20	0.2663
0.30	0.2181
0.40	0.1759
0.50	0.1209
0.60	0.0839
0.70	0.0518
0.80	0.0275
0.90	0.0121
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1522

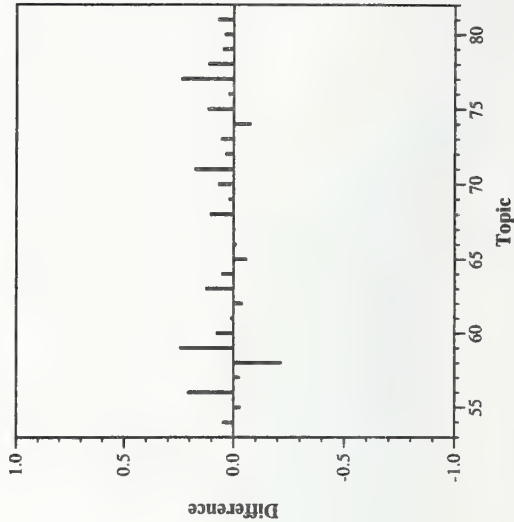
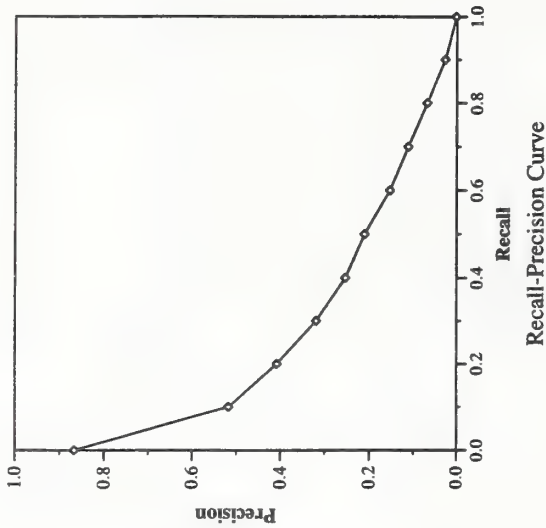
Document Level Averages	
	Precision
At 5 docs	0.3643
At 10 docs	0.3143
At 15 docs	0.3024
At 20 docs	0.2964
At 30 docs	0.2810
At 100 docs	0.2036
At 200 docs	0.1429
At 500 docs	0.0841
At 1000 docs	0.0507
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2125



Summary Statistics		
Run Number	tno8dis	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1621	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8672
0.10	0.5174
0.20	0.4078
0.30	0.3193
0.40	0.2538
0.50	0.2109
0.60	0.1539
0.70	0.1125
0.80	0.0692
0.90	0.0276
1.00	0.0018
Average precision over all relevant docs	
non-interpolated	0.2407

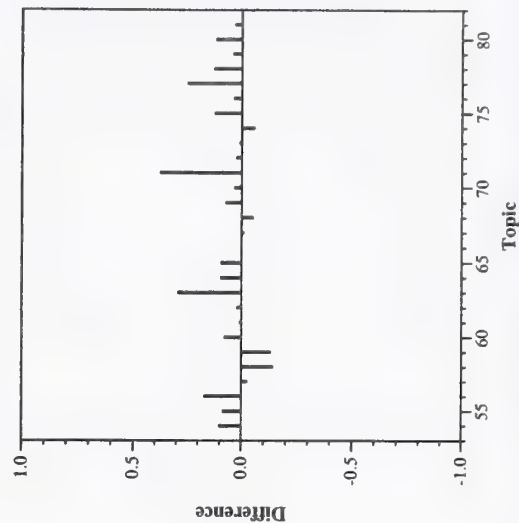
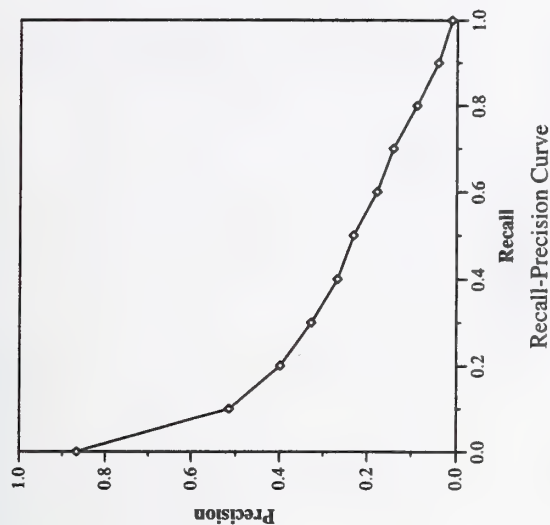
Document Level Averages	
	Precision
At 5 docs	0.5786
At 10 docs	0.5107
At 15 docs	0.4714
At 20 docs	0.4196
At 30 docs	0.3905
At 100 docs	0.2575
At 200 docs	0.1868
At 500 docs	0.1026
At 1000 docs	0.0579
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2844



Summary Statistics		
Run Number	tno8dpx	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1725	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8665
0.10	0.5148
0.20	0.3984
0.30	0.3270
0.40	0.2678
0.50	0.2315
0.60	0.1788
0.70	0.1421
0.80	0.0892
0.90	0.0412
1.00	0.0114
Average precision over all relevant docs	
non-interpolated	0.2523

Document Level Averages	
	Precision
At 5 docs	0.5500
At 10 docs	0.5000
At 15 docs	0.4738
At 20 docs	0.4446
At 30 docs	0.4107
At 100 docs	0.2725
At 200 docs	0.1971
At 500 docs	0.1077
At 1000 docs	0.0616
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2881



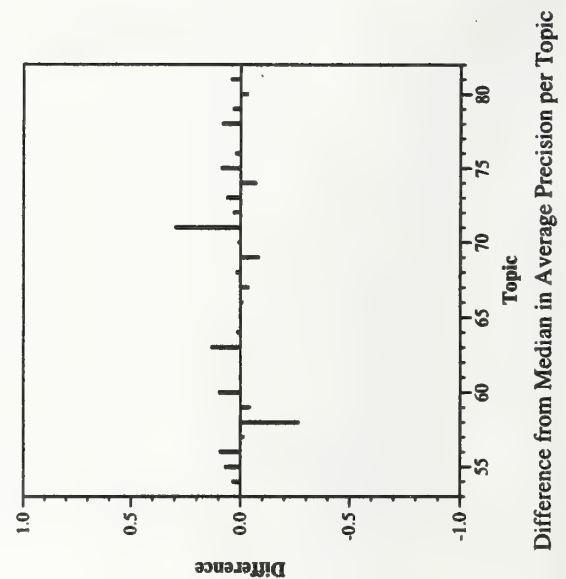
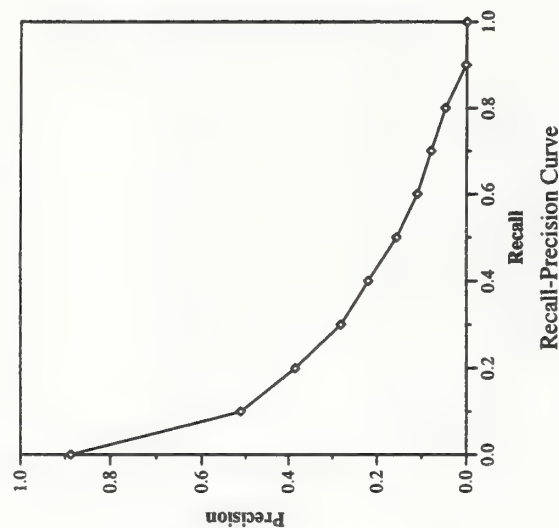
Difference from Median in Average Precision per Topic

Cross-language track results — TwentyOne

Summary Statistics		
Run Number	tno8gr	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1414	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8872
0.10	0.5114
0.20	0.3844
0.30	0.2810
0.40	0.2200
0.50	0.1559
0.60	0.1097
0.70	0.0790
0.80	0.0472
0.90	0.0021
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.2111

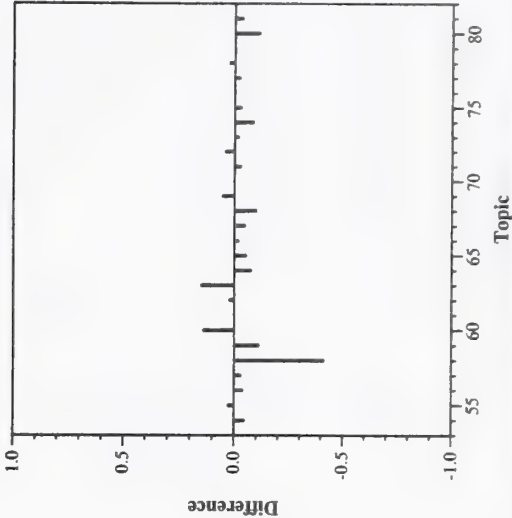
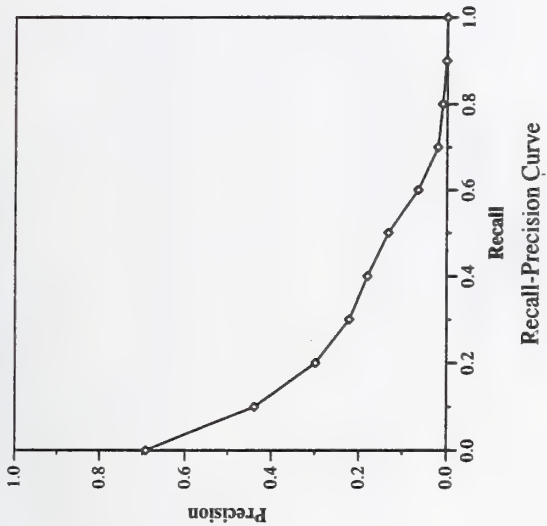
Document Level Averages	
	Precision
At 5 docs	0.5286
At 10 docs	0.4929
At 15 docs	0.4571
At 20 docs	0.4054
At 30 docs	0.3726
At 100 docs	0.2561
At 200 docs	0.1771
At 500 docs	0.0896
At 1000 docs	0.0505
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2704



Summary Statistics		
Run Number	umd99b1	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1208	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6926
0.10	0.4388
0.20	0.2983
0.30	0.2217
0.40	0.1807
0.50	0.1327
0.60	0.0653
0.70	0.0204
0.80	0.0105
0.90	0.0017
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1616

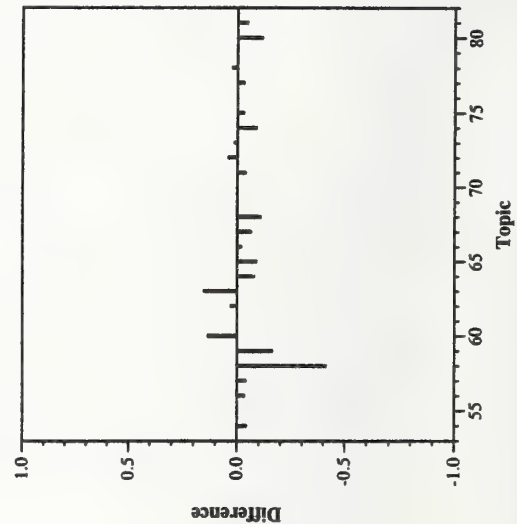
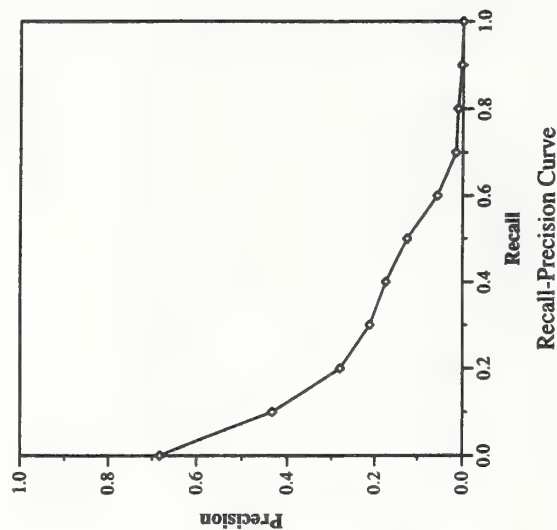
Document Level Averages	
	Precision
At 5 docs	0.4857
At 10 docs	0.4286
At 15 docs	0.3976
At 20 docs	0.3768
At 30 docs	0.3262
At 100 docs	0.2125
At 200 docs	0.1425
At 500 docs	0.0738
At 1000 docs	0.0431
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2124



Summary Statistics		
Run Number	umd99b2	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1127	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6830
0.10	0.4333
0.20	0.2806
0.30	0.2116
0.40	0.1741
0.50	0.1242
0.60	0.0562
0.70	0.0152
0.80	0.0109
0.90	0.0027
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1555

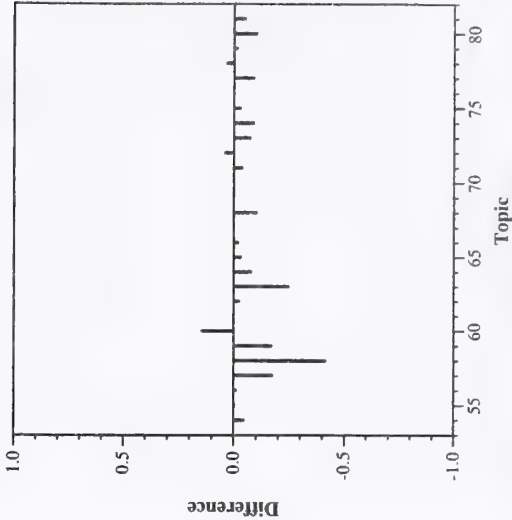
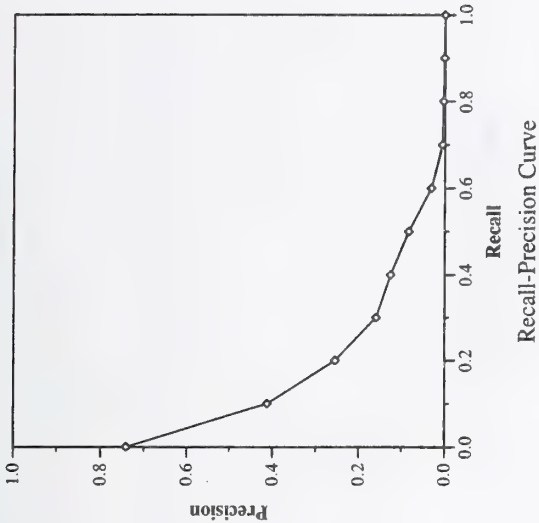
Document Level Averages	
	Precision
At 5 docs	0.4714
At 10 docs	0.4179
At 15 docs	0.3905
At 20 docs	0.3643
At 30 docs	0.3190
At 100 docs	0.2061
At 200 docs	0.1391
At 500 docs	0.0719
At 1000 docs	0.0402
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2060



Summary Statistics		
Run Number	umd99b3	
Run Description	english, automatic, T+D+N	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1107	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7416
0.10	0.4121
0.20	0.2547
0.30	0.1599
0.40	0.1257
0.50	0.0836
0.60	0.0307
0.70	0.0059
0.80	0.0033
0.90	0.0015
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1344

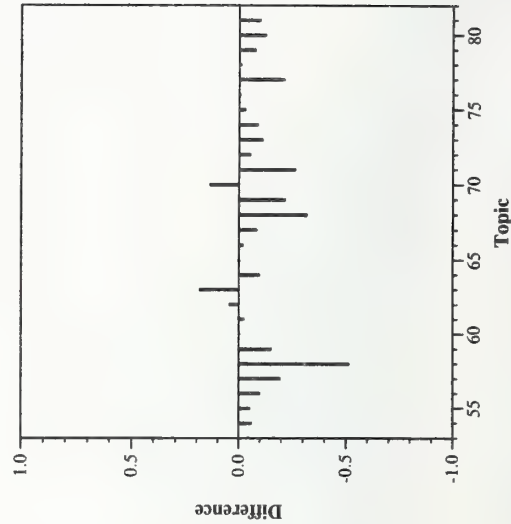
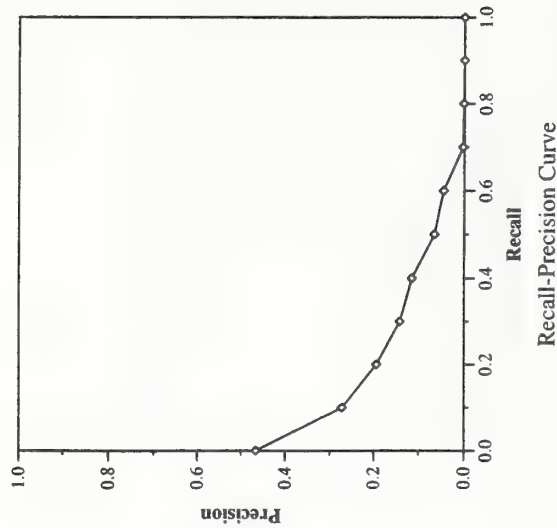
Document Level Averages	
	Precision
At 5 docs	0.4786
At 10 docs	0.4107
At 15 docs	0.3833
At 20 docs	0.3554
At 30 docs	0.3048
At 100 docs	0.1818
At 200 docs	0.1252
At 500 docs	0.0641
At 1000 docs	0.0395
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1827



Summary Statistics		
Run Number	umd99c1	
Run Description	english, automatic, T	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	955	

Recall Level Precision Averages	
Recall	Precision
0.00	0.4661
0.10	0.2719
0.20	0.1936
0.30	0.1411
0.40	0.1145
0.50	0.0650
0.60	0.0452
0.70	0.0026
0.80	0.0015
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1003

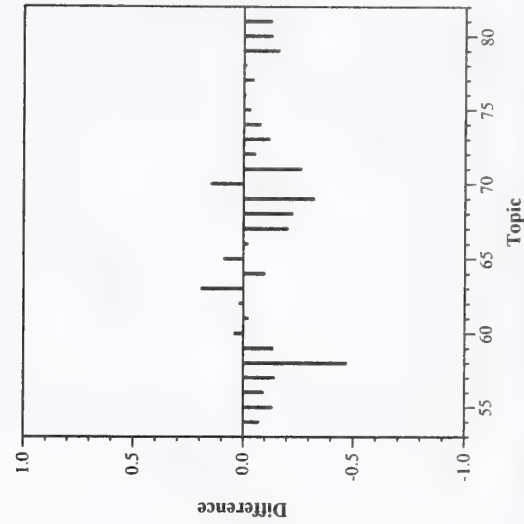
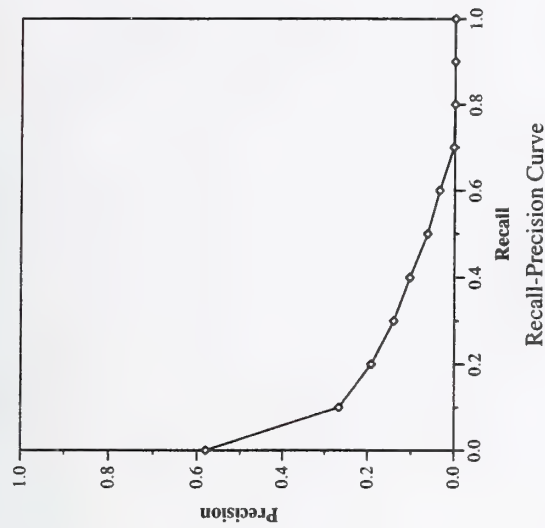
Document Level Averages	
	Precision
At 5 docs	0.2929
At 10 docs	0.2786
At 15 docs	0.2595
At 20 docs	0.2536
At 30 docs	0.2298
At 100 docs	0.1461
At 200 docs	0.1038
At 500 docs	0.0604
At 1000 docs	0.0341
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1478



Summary Statistics		
Run Number	umd99c2	
Run Description	english, automatic, T	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	832	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5800
0.10	0.2673
0.20	0.1921
0.30	0.1407
0.40	0.1033
0.50	0.0622
0.60	0.0345
0.70	0.0022
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1030

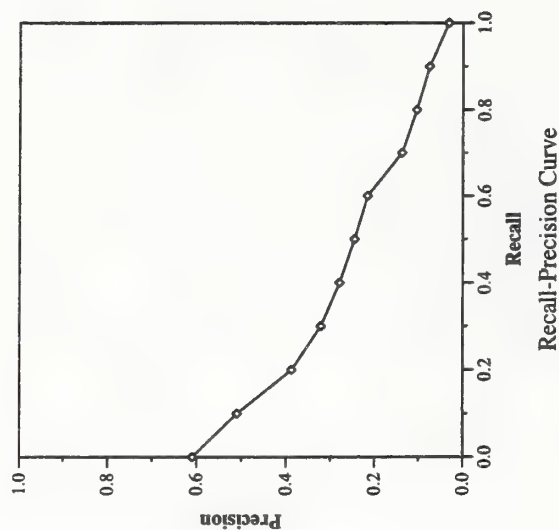
Document Level Averages	
	Precision
At 5 docs	0.3714
At 10 docs	0.2893
At 15 docs	0.2762
At 20 docs	0.2643
At 30 docs	0.2345
At 100 docs	0.1439
At 200 docs	0.0954
At 500 docs	0.0501
At 1000 docs	0.0297
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1473



Summary Statistics			
Run Number	sle2Ef1		
Run Description	Italian topics against English docs		
Number of Topics	24		
Total number of documents over all topics			
Retrieved:	24000		
Relevant:	956		
Rel-ret:	709		

Recall Level Precision Averages	
Recall	Precision
0.00	0.6093
0.10	0.5099
0.20	0.3871
0.30	0.3213
0.40	0.2794
0.50	0.2450
0.60	0.2160
0.70	0.1376
0.80	0.1043
0.90	0.0750
1.00	0.0316
Average precision over all relevant docs	
non-interpolated	0.2514

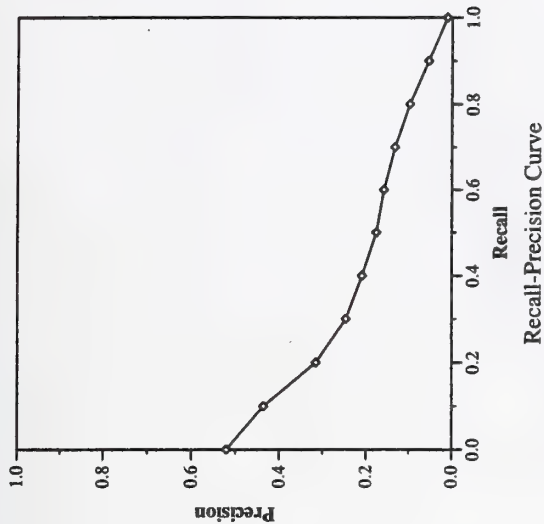
Document Level Averages	
	Precision
At 5 docs	0.4083
At 10 docs	0.3458
At 15 docs	0.3194
At 20 docs	0.3021
At 30 docs	0.2653
At 100 docs	0.1721
At 200 docs	0.1112
At 500 docs	0.0545
At 1000 docs	0.0295
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2590



Summary Statistics			
Run Number	sl12Etd1		
Run Description	Italian topics against English docs		
Number of Topics	24		
Total number of documents over all topics			
Retrieved:	24000		
Relevant:	956		
Rel-ret:	653		

Recall Level Precision Averages	
Recall	Precision
0.00	0.5205
0.10	0.4361
0.20	0.3159
0.30	0.2470
0.40	0.2096
0.50	0.1764
0.60	0.1591
0.70	0.1330
0.80	0.0984
0.90	0.0542
1.00	0.0112
Average precision over all relevant docs	
non-interpolated	0.2016

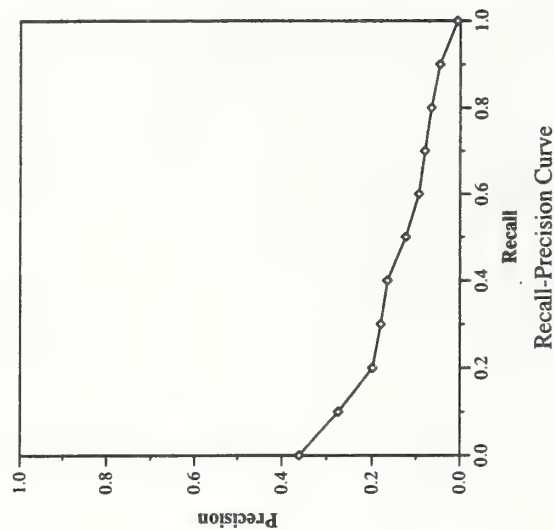
Document Level Averages	
At 5 docs	0.3667
At 10 docs	0.2833
At 15 docs	0.2583
At 20 docs	0.2354
At 30 docs	0.2139
At 100 docs	0.1475
At 200 docs	0.1017
At 500 docs	0.0497
At 1000 docs	0.0272
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2124



Summary Statistics			
Run Number	sle12Et1		
Run Description	Italian topics docs	against English	
Number of Topics	24		
Total number of documents over all topics			
Retrieved:	23154		
Relevant:	956		
Rel-ret:	532		

Recall Level Precision Averages	
Recall	Precision
0.00	0.3608
0.10	0.2745
0.20	0.1991
0.30	0.1804
0.40	0.1654
0.50	0.1239
0.60	0.0940
0.70	0.0805
0.80	0.0659
0.90	0.0461
1.00	0.0062
Average precision over all relevant docs	
non-interpolated	0.1316

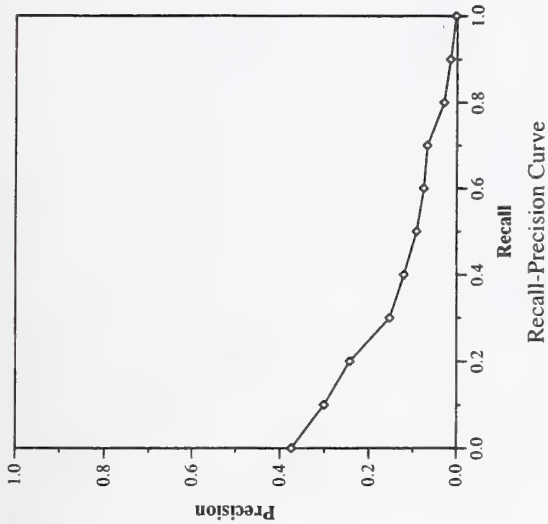
Document Level Averages	
	Precision
At 5 docs	0.2167
At 10 docs	0.2000
At 15 docs	0.1889
At 20 docs	0.1771
At 30 docs	0.1486
At 100 docs	0.0987
At 200 docs	0.0748
At 500 docs	0.0399
At 1000 docs	0.0222
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1498



Summary Statistics			
Run Number	TW8F2E		
Run Description	French topics against English docs		
Number of Topics	24		
Total number of documents over all topics			
Retrieved:	24000		
Relevant:	956		
Rel-ret:	557		

Recall Level Precision Averages	
Recall	Precision
0.00	0.3740
0.10	0.3010
0.20	0.2431
0.30	0.1531
0.40	0.1208
0.50	0.0919
0.60	0.0757
0.70	0.0678
0.80	0.0297
0.90	0.0151
1.00	0.0025
Average precision over all relevant docs	
non-interpolated	0.1179

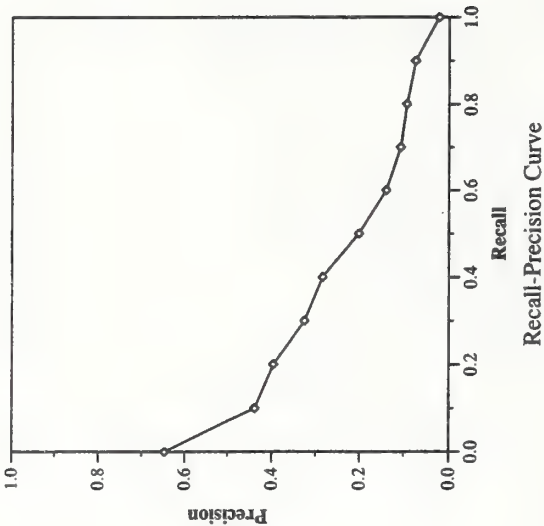
Document Level Averages	
	Precision
At 5 docs	0.2000
At 10 docs	0.2250
At 15 docs	0.2111
At 20 docs	0.1938
At 30 docs	0.1736
At 100 docs	0.0954
At 200 docs	0.0692
At 500 docs	0.0389
At 1000 docs	0.0232
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1365



Summary Statistics			
Run Number	TW8E2F		
Run Description	English topics docs	against French docs	
Number of Topics	27		
Total number of documents over all topics			
Retrieved:	27000		
Relevant:	578		
Rel-ret:	402		

Recall Level Precision Averages	
Recall	Precision
0.00	0.6471
0.10	0.4387
0.20	0.3965
0.30	0.3256
0.40	0.2844
0.50	0.2014
0.60	0.1399
0.70	0.1070
0.80	0.0935
0.90	0.0742
1.00	0.0220
Average precision over all relevant docs	
non-interpolated	0.2291

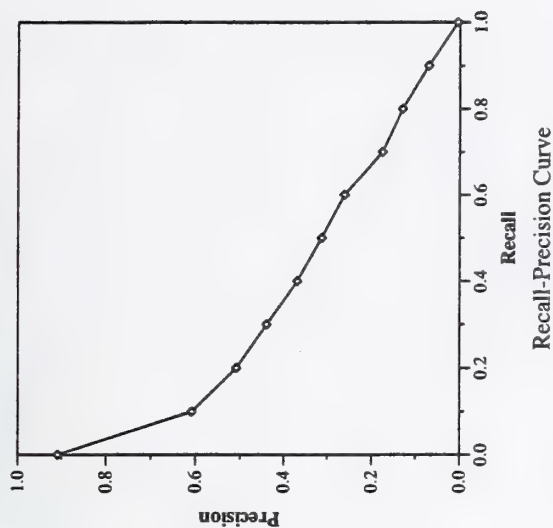
Document Level Averages	
	Precision
At 5 docs	0.3852
At 10 docs	0.3000
At 15 docs	0.2593
At 20 docs	0.2167
At 30 docs	0.1728
At 100 docs	0.0859
At 200 docs	0.0526
At 500 docs	0.0252
At 1000 docs	0.0149
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2327



Summary Statistics		
Run Number	tno8mx	
Run Description	Merged monolingual run	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	2421	
Rel-ret:	1977	

Recall Level Precision Averages	
Recall	Precision
0.00	0.9088
0.10	0.6078
0.20	0.5087
0.30	0.4397
0.40	0.3683
0.50	0.3126
0.60	0.2609
0.70	0.1751
0.80	0.1296
0.90	0.0703
1.00	0.0045
Average precision over all relevant docs	
non-interpolated	0.3225

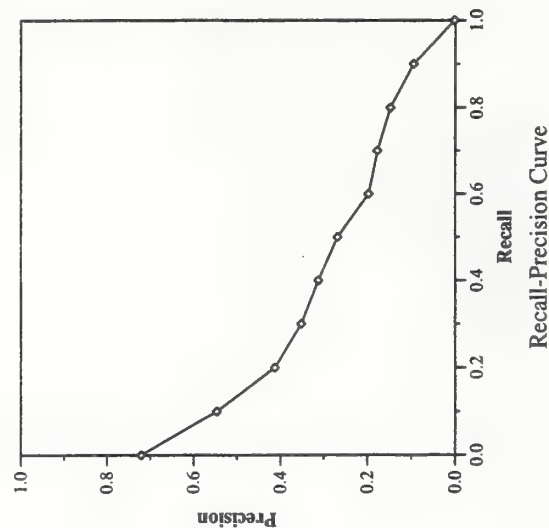
Document Level Averages	
	Precision
At 5 docs	0.6429
At 10 docs	0.6036
At 15 docs	0.5714
At 20 docs	0.5018
At 30 docs	0.4631
At 100 docs	0.3321
At 200 docs	0.2327
At 500 docs	0.1254
At 1000 docs	0.0706
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3592



Summary Statistics		
Run Number	RaliWebE2EF	
Run Description	English topics against English, French docs	
Number of Topics	28	
Total number of documents over all topics	28000	
Retrieved:	1534	
Relevant:	1160	
Rel-ret:		

Recall Level Precision Averages		
	Recall	Precision
Average precision over all relevant docs	0.00	0.7209
	0.10	0.5467
	0.20	0.4130
	0.30	0.3521
	0.40	0.3126
	0.50	0.2678
	0.60	0.1975
	0.70	0.1774
	0.80	0.1482
	0.90	0.0954
	1.00	0.0017
Average precision over all relevant docs		
non-interpolated		0.2744

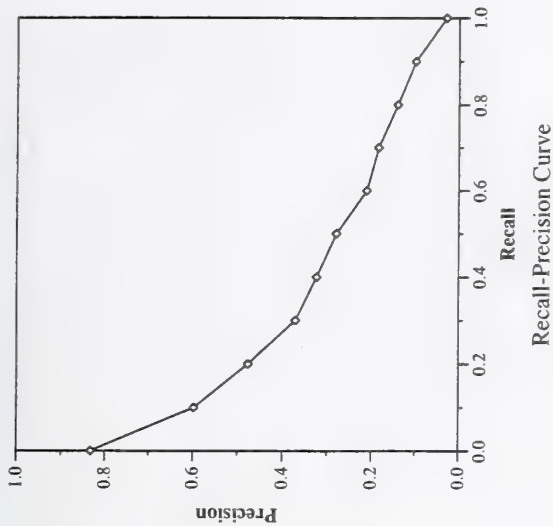
Document Level Averages	
	Precision
At 5 docs	0.4786
At 10 docs	0.4357
At 15 docs	0.4190
At 20 docs	0.3732
At 30 docs	0.3238
At 100 docs	0.2143
At 200 docs	0.1486
At 500 docs	0.0760
At 1000 docs	0.0414
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2974



Summary Statistics		
Run Number	RaliWebF2EF	
Run Description	French topics against English, French docs	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	1534	
Rel-ret:	1237	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8322
0.10	0.5974
0.20	0.4762
0.30	0.3709
0.40	0.3237
0.50	0.2789
0.60	0.2109
0.70	0.1840
0.80	0.1407
0.90	0.0993
1.00	0.0287
Average precision over all relevant docs	
non-interpolated	0.3012

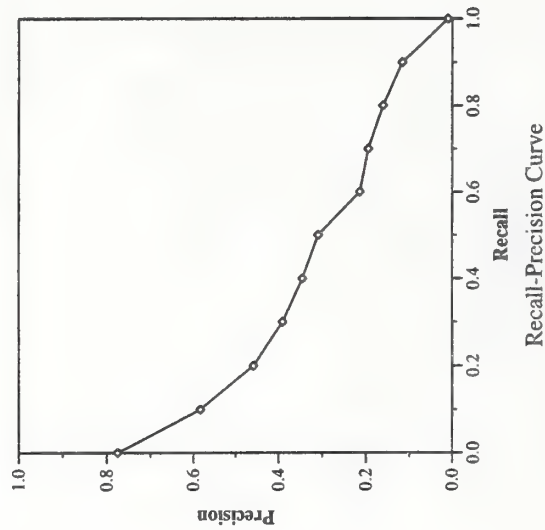
Document Level Averages	
	Precision
At 5 docs	0.5071
At 10 docs	0.4429
At 15 docs	0.4214
At 20 docs	0.3857
At 30 docs	0.3476
At 100 docs	0.2143
At 200 docs	0.1514
At 500 docs	0.0783
At 1000 docs	0.0442
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3305



Summary Statistics		
Run Number	RaliHanE2EF	
Run Description	English topics against English, French docs	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	1534	
Rel-ret:	1161	

Recall Level Precision Averages		
Recall	Precision	
0.00	0.7750	
0.10	0.5838	
0.20	0.4582	
0.30	0.3907	
0.40	0.3452	
0.50	0.3088	
0.60	0.2139	
0.70	0.1948	
0.80	0.1608	
0.90	0.1163	
1.00	0.0103	
Average precision over all relevant docs		
non-interpolated	0.3027	

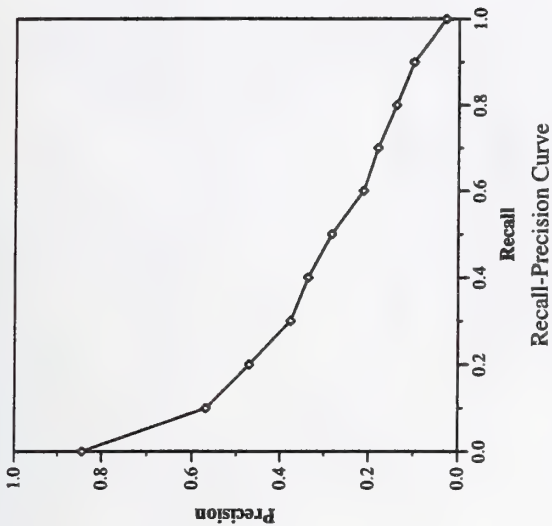
Document Level Averages	
	Precision
At 5 docs	0.4929
At 10 docs	0.4429
At 15 docs	0.4095
At 20 docs	0.3786
At 30 docs	0.3226
At 100 docs	0.2182
At 200 docs	0.1500
At 500 docs	0.0764
At 1000 docs	0.0415
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3245



Summary Statistics		
Run Number	RaliHanF2EF	
Run Description	French topics against English, French docs	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	1534	
Rel-ret:	1253	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8440
0.10	0.5693
0.20	0.4710
0.30	0.3766
0.40	0.3372
0.50	0.2835
0.60	0.2120
0.70	0.1798
0.80	0.1390
0.90	0.1003
1.00	0.0290
Average precision over all relevant docs	
non-interpolated	0.3002

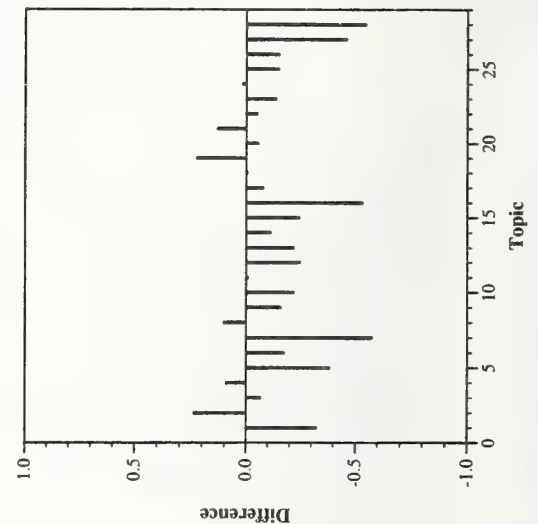
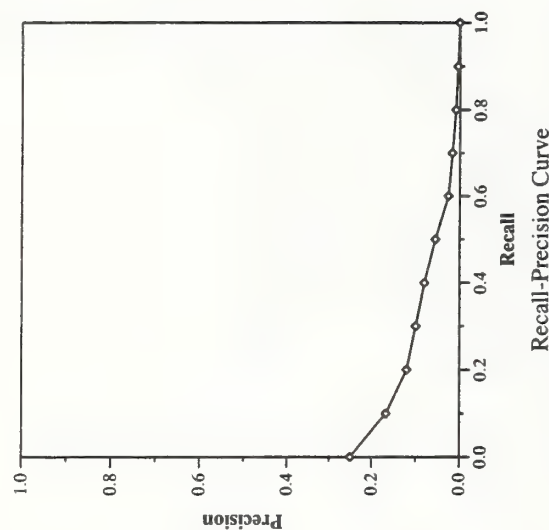
Document Level Averages	
	Precision
At 5 docs	0.5286
At 10 docs	0.4571
At 15 docs	0.4167
At 20 docs	0.3768
At 30 docs	0.3369
At 100 docs	0.2150
At 200 docs	0.1507
At 500 docs	0.0784
At 1000 docs	0.0448
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3286



Summary Statistics		
Run Number	EIT99geg	
Run Description	English topics, automatic	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	27138	
Relevant:	1294	
Rel-ret:	542	

Recall Level Precision Averages	
Recall	Precision
0.00	0.2501
0.10	0.1671
0.20	0.1202
0.30	0.1000
0.40	0.0806
0.50	0.0556
0.60	0.0261
0.70	0.0173
0.80	0.0095
0.90	0.0048
1.00	0.0010
Average precision over all relevant docs	
non-interpolated	0.0624

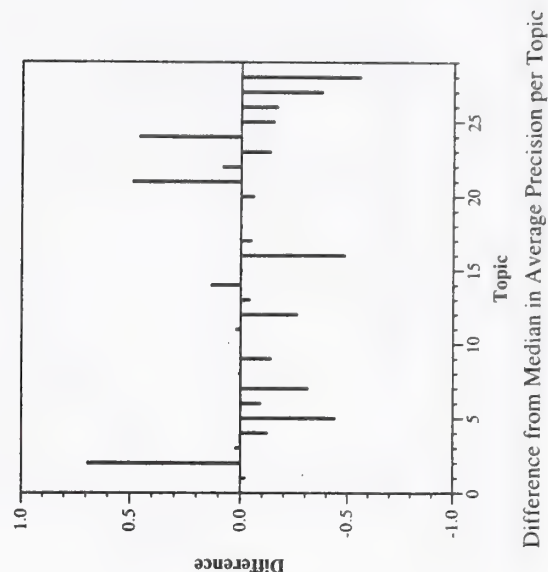
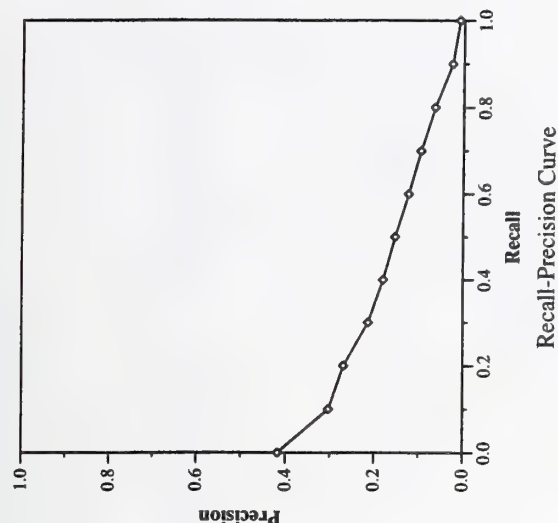
Document Level Averages	
	Precision
At 5 docs	0.1143
At 10 docs	0.0893
At 15 docs	0.0905
At 20 docs	0.0911
At 30 docs	0.0964
At 100 docs	0.0721
At 200 docs	0.0480
At 500 docs	0.0290
At 1000 docs	0.0194
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1002



Summary Statistics		
Run Number	EIT99fg	
Run Description	French topics, automatic	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	27031	
Relevant:	1294	
Rel-ret:	827	

Recall Level Precision Averages	
Recall	Precision
0.00	0.4177
0.10	0.3032
0.20	0.2690
0.30	0.2135
0.40	0.1797
0.50	0.1530
0.60	0.1232
0.70	0.0959
0.80	0.0648
0.90	0.0257
1.00	0.0095
Average precision over all relevant docs	
non-interpolated	0.1547

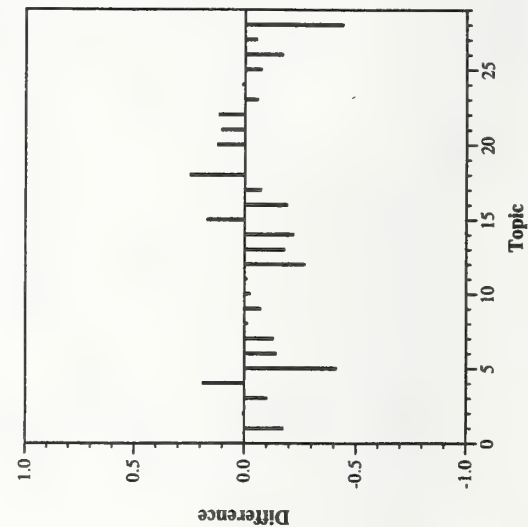
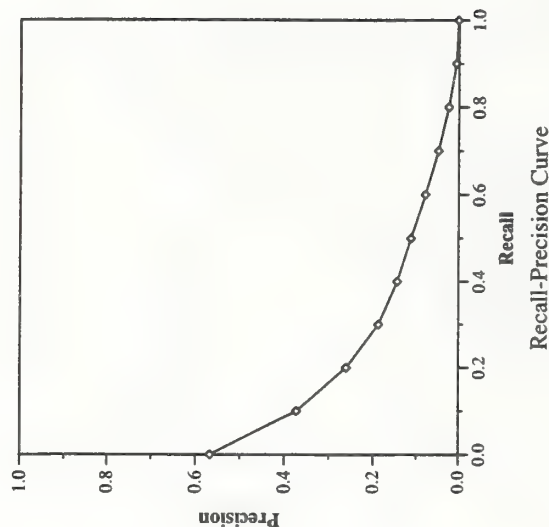
Document Level Averages	
	Precision
At 5 docs	0.2500
At 10 docs	0.2286
At 15 docs	0.2286
At 20 docs	0.2143
At 30 docs	0.1917
At 100 docs	0.1386
At 200 docs	0.0889
At 500 docs	0.0495
At 1000 docs	0.0295
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1844



Summary Statistics		
Run Number	EIT99gmt	
Run Description	French topics, automatic	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	1294	
Rel-ret:	787	

Recall Level Precision Averages	
Recall	Precision
0.00	0.5672
0.10	0.3732
0.20	0.2615
0.30	0.1882
0.40	0.1452
0.50	0.1133
0.60	0.0795
0.70	0.0483
0.80	0.0244
0.90	0.0064
1.00	0.0012
Average precision over all relevant docs	
non-interpolated	0.1438

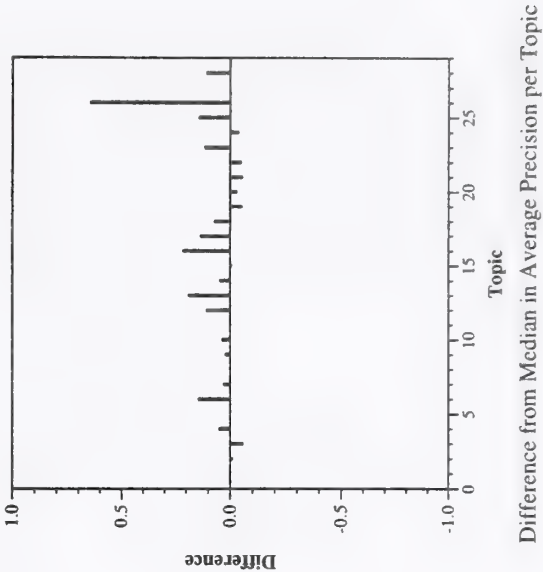
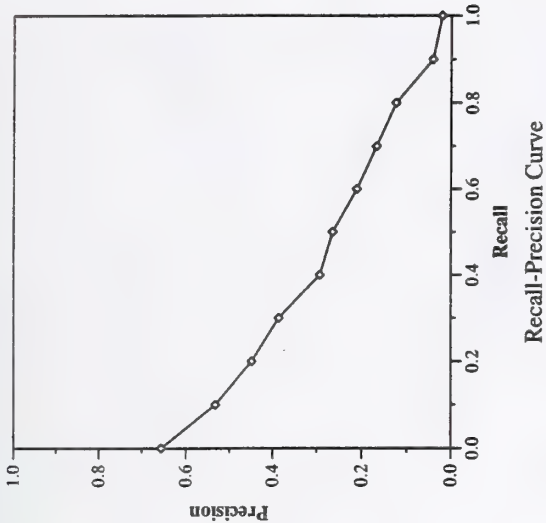
Document Level Averages	
	Precision
At 5 docs	0.3071
At 10 docs	0.2893
At 15 docs	0.2405
At 20 docs	0.2268
At 30 docs	0.2083
At 100 docs	0.1414
At 200 docs	0.0905
At 500 docs	0.0467
At 1000 docs	0.0281
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1965



Summary Statistics	
Run Number	BKCLGR01
Run Description	English topics, automatic
Number of Topics	28
Total number of documents over all topics	
Retrieved:	28000
Relevant:	1294
Rel-ret:	907

Recall Level Precision Averages	
Recall	Precision
0.00	0.6564
0.10	0.5326
0.20	0.4485
0.30	0.3858
0.40	0.2924
0.50	0.2640
0.60	0.2107
0.70	0.1674
0.80	0.1242
0.90	0.0412
1.00	0.0208
Average precision over all relevant docs	
non-interpolated	0.2707

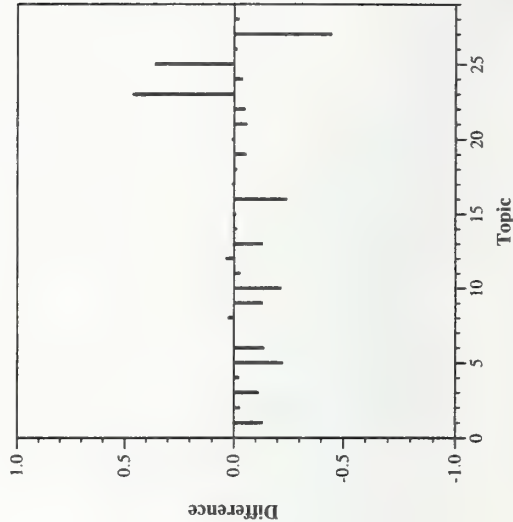
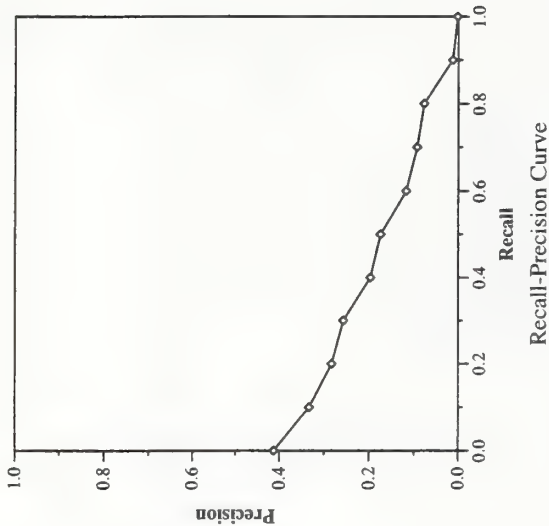
Document Level Averages	
	Precision
At 5 docs	0.4714
At 10 docs	0.4000
At 15 docs	0.3762
At 20 docs	0.3500
At 30 docs	0.3060
At 100 docs	0.2168
At 200 docs	0.1311
At 500 docs	0.0605
At 1000 docs	0.0324
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2752



Summary Statistics		
Run Number	BKCLGR02	
Run Description	English topics, automatic	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	1294	
Rel-ret:	733	

Recall Level Precision Averages	
Recall	Precision
0.00	0.4124
0.10	0.3329
0.20	0.2826
0.30	0.2568
0.40	0.1971
0.50	0.1746
0.60	0.1175
0.70	0.0935
0.80	0.0774
0.90	0.0131
1.00	0.0018
Average precision over all relevant docs	
non-interpolated	0.1667

Document Level Averages	
	Precision
At 5 docs	0.2929
At 10 docs	0.2429
At 15 docs	0.2310
At 20 docs	0.2179
At 30 docs	0.1988
At 100 docs	0.1529
At 200 docs	0.1002
At 500 docs	0.0485
At 1000 docs	0.0262
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1808

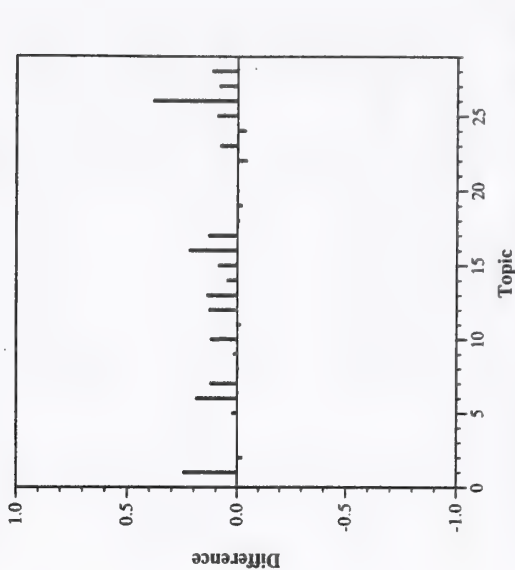
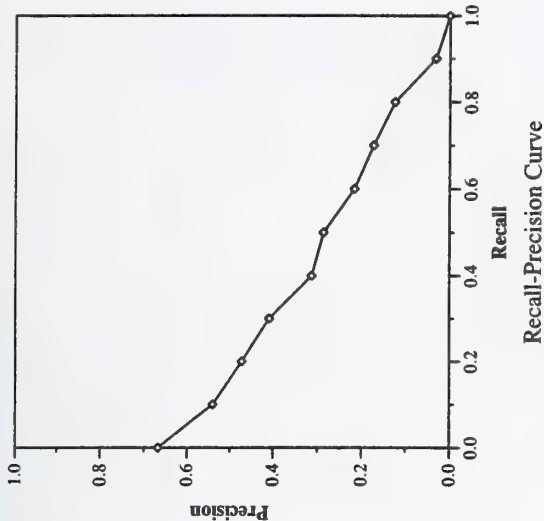


Difference from Median in Average Precision per Topic

Summary Statistics		
Run Number	BKCLGR03	
Run Description	English topics, automatic	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	1294	
Rel-ret:	936	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6671
0.10	0.5402
0.20	0.4721
0.30	0.4087
0.40	0.3131
0.50	0.2864
0.60	0.2163
0.70	0.1726
0.80	0.1241
0.90	0.0314
1.00	0.0008
Average precision over all relevant docs	
non-interpolated	0.2832

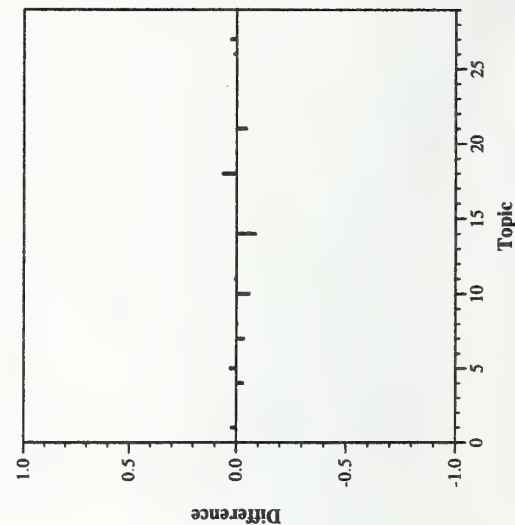
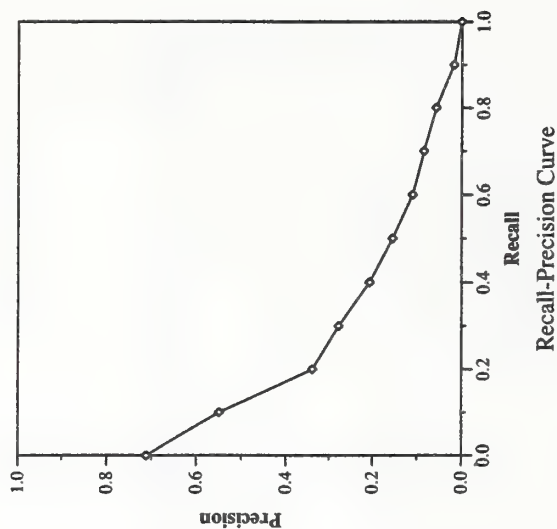
Document Level Averages	
	Precision
At 5 docs	0.4786
At 10 docs	0.4357
At 15 docs	0.3929
At 20 docs	0.3571
At 30 docs	0.3286
At 100 docs	0.2296
At 200 docs	0.1386
At 500 docs	0.0636
At 1000 docs	0.0334
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3064



Summary Statistics		
Run Number	BKCLGR04	
Run Description	English topics, automatic	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	1294	
Rel-ret:	890	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7111
0.10	0.5492
0.20	0.3388
0.30	0.2774
0.40	0.2068
0.50	0.1550
0.60	0.1103
0.70	0.0848
0.80	0.0572
0.90	0.0178
1.00	0.0008
Average precision over all relevant docs	
non-interpolated	0.2049

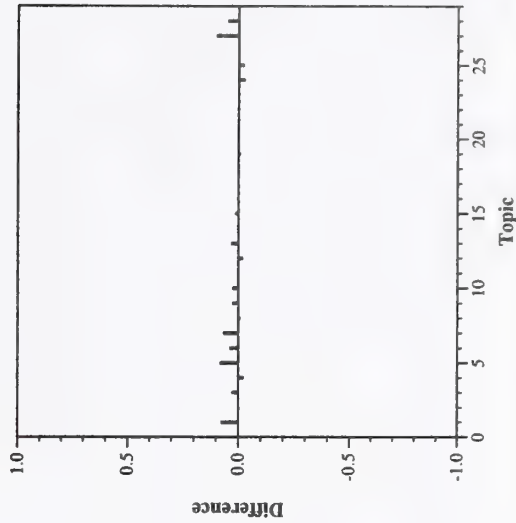
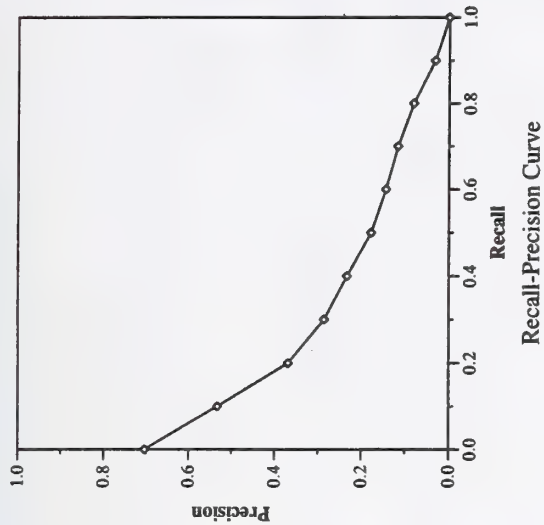
Document Level Averages	
	Precision
At 5 docs	0.4429
At 10 docs	0.4000
At 15 docs	0.3405
At 20 docs	0.3232
At 30 docs	0.2881
At 100 docs	0.1786
At 200 docs	0.1100
At 500 docs	0.0574
At 1000 docs	0.0318
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2487



Summary Statistics		
Run Number	BKCLGR05	
Run Description	English topics, automatic	
Number of Topics	28	
Total number of documents over all topics		
Retrieved:	28000	
Relevant:	1294	
Rel-ret:	921	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7036
0.10	0.5325
0.20	0.3689
0.30	0.2854
0.40	0.2327
0.50	0.1768
0.60	0.1435
0.70	0.1159
0.80	0.0805
0.90	0.0319
1.00	0.0004
Average precision over all relevant docs	
non-interpolated	0.2232

Document Level Averages	
	Precision
At 5 docs	0.4429
At 10 docs	0.3893
At 15 docs	0.3690
At 20 docs	0.3446
At 30 docs	0.3143
At 100 docs	0.1932
At 200 docs	0.1232
At 500 docs	0.0612
At 1000 docs	0.0329
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2608



TREC-8 Adaptive Filtering Results, LF1 Measure, Years 92-94 (Part 1)

Topic	A	B	C	D	E	F	G	H	I	J	Maximal
351	28	22	28	21	-14	-14	-2	-3	-949	-957	84
352	-7	-9	-7	-7	-5	-5	-2	-2	-132	-132	741
353	-1	-2	-1	-1	9	12	-2	-2	-611	-613	132
354	-1	4	-1	-1	-5	-5	-2	-18	-1035	-1035	213
355	-8	-14	-8	-8	-20	-20	-2	-2	-142	-142	6
356	-17	0	-13	-13	0	0	-2	-2	-184	-184	69
357	-5	-11	2	0	40	39	-2	-1	-296	-298	255
358	-22	-6	-20	-20	-8	-8	-2	-2	-256	-256	6
359	-16	-12	-12	-12	-97	-97	-2	-509	-365	-369	114
360	-8	-8	-8	-8	-11	-11	-6	-77	-462	-461	42
361	-16	-24	-16	-16	-36	-36	-2	-10	-490	-490	6
362	-13	-14	-13	-13	-4	-4	-2	-2	-249	-251	15
363	-10	-3	-8	-8	-114	-114	-2	-57	-2553	-2563	18
364	-9	-6	-7	-7	-29	-29	1	-1	-15	-15	9
365	27	7	27	23	31	31	2	4	-47	-47	33
366	-3	-5	10	32	3	11	-2	-6	-227	-229	63
367	-3	-10	-3	6	9	9	-2	-8	-895	-903	126
368	-12	1	-10	-10	2	2	-2	-12	-75	-75	15
369	-9	-9	-9	-15	-8	-8	-2	-2	-597	-597	3
370	15	6	-1	2	0	0	-2	-3	-384	-395	159
371	-12	-8	-8	-8	-154	-154	-2	-40	-2768	-2772	6
372	-7	-18	-5	-5	-12	-10	-6	-2	-229	-229	45
373	-19	-8	-15	-15	-27	-27	12	-2	-23	-23	42
374	-5	-1	-5	-5	-17	-17	0	-3	-350	-350	234
375	-9	21	-9	-9	-15	-11	2	-14	-250	-252	60
376	-5	-5	-5	-5	-103	-121	-2	-87	-2050	-2038	90
377	-8	-1	-8	-8	-29	-29	-2	-2	-148	-150	45
378	-14	-14	-10	-10	-6	-4	-2	-2	-2165	-2249	285
379	-20	-8	-16	-16	0	0	-2	-2	-536	-536	0
380	-2	-12	-2	-2	-109	-109	-2	-4	-101	-103	3
381	-14	-10	-12	-12	-129	-129	-2	-2	-538	-538	21
382	1	-7	1	-1	0	2	2	-16	-474	-476	18
383	-13	-10	-11	-13	-131	-131	-6	-352	-1320	-1486	246
384	-18	-10	-14	-14	-17	-17	-2	-4	-1057	-1063	12
385	-8	-7	-8	-8	6	6	-2	-10	-857	-873	108
386	-20	-12	-16	-16	-13	-13	-2	-40	-545	-545	21
387	-9	-6	-9	-7	-21	-24	-4	-96	-169	-168	39
388	-1	-3	-1	-1	0	0	-2	-2	-105	-105	51
389	-10	-8	-8	-8	-2	-2	-2	-6	-387	-387	510
390	-3	-3	2	4	-2	-2	3	-183	-777	-782	213
391	46	-6	82	84	1	8	-3	-305	-3475	-4210	549
392	-4	3	-4	2	6	10	13	-15	-165	-165	192
393	-18	-14	-14	-14	-11	-11	-2	-2	-199	-199	27
394	-5	-10	-5	-5	-157	-157	-40	-66	-702	-702	15
395	2	-3	5	-13	0	0	13	-16	-1809	-2074	312
396	3	2	0	3	1	3	9	-9	-553	-575	33
397	-8	-7	-6	-6	-74	-74	-6	-18	-290	-290	21
398	14	20	14	14	19	19	-2	-18	-3030	-3146	60
399	-5	-5	-5	-1	2	2	-2	-4	-810	-810	39
400	48	16	48	50	-16	-16	9	-8	-797	-801	150

KEY

A	CL99afL1b
B	CL99afL1a
C	CL99afL1c
D	CL99afL1d
E	pir9LF1
F	pir9LF1a
G	IOWAF992
H	IOWAF993
I	kdd8f003
J	kdd8f001

TREC-8 Adaptive Filtering Results, LF1 Measure, Years 92-94 (Part 2)

Topic	K	L	M	N	O	P	Q	R	S	Maximal
351	-1198	-16	15	13	-93	16	14	-36	-87	84
352	-132	11	21	19	-1002	-6	-11	-23	-70	741
353	-603	-40	-9	-9	-253	18	24	-20	-29	132
354	-1698	-29	-5	-4	-29	1	9	0	0	213
355	-142	-20	-4	-6	-436	-20	-20	-20	-32	6
356	-184	-24	0	0	-322	4	2	12	12	69
357	-294	-8	6	4	-84	-10	-10	-17	-26	255
358	-256	-88	-34	-40	-772	-28	-28	-6	-6	6
359	-365	-137	-56	-56	-622	-29	-29	-8	-8	114
360	-462	-97	-20	-24	-878	-17	-17	3	3	42
361	-490	-186	-68	-68	-1002	-20	-20	-20	-32	6
362	-247	-148	-54	-54	-688	-16	-16	-20	-26	15
363	-2629	-182	-76	-78	-358	-44	-44	-20	-32	18
364	-15	-29	-11	-17	-161	-13	-13	-17	-29	9
365	-65	-2	12	12	-1002	25	25	29	29	33
366	-233	15	37	35	-1002	43	45	16	16	63
367	-983	-47	-31	-47	-62	1	7	-20	-48	126
368	-73	-24	-7	-9	-104	-7	-10	-17	-9	15
369	-603	-6	-4	-4	-6	-14	-14	-17	-29	3
370	-355	-16	8	6	-202	4	3	3	3	159
371	-2760	-122	-48	-53	-1002	-22	-22	-20	-32	6
372	-229	-39	6	2	-974	-4	-4	7	5	45
373	-23	-67	-14	-20	-285	-15	-15	-17	-29	42
374	-322	-13	30	28	-44	15	24	48	48	234
375	-262	-30	-14	-8	-1002	-10	-12	8	8	60
376	-2041	-162	-56	-62	-442	-24	-26	11	11	90
377	-155	-56	-14	-22	-276	-9	-12	-14	-28	45
378	-2232	-172	-56	-61	-1002	-41	-33	-20	-32	285
379	-536	-62	-18	-20	-658	-22	-22	-16	-16	0
380	-119	-123	-43	-47	-1002	-13	-13	-20	-29	3
381	-538	-115	-51	-57	-944	-30	-30	-20	-32	21
382	-486	-32	-4	-8	-533	-2	-2	-17	-29	18
383	-1334	-106	-26	-23	-787	-13	-13	-12	-20	246
384	-1609	-113	-30	-36	-93	-18	-18	-20	-32	12
385	-871	-47	-6	-8	-553	-2	4	-17	-13	108
386	-539	-81	-21	-27	-573	-12	-12	-17	-29	21
387	-173	-129	-53	-51	-323	-18	-21	-17	-26	39
388	-105	-28	1	2	-69	-18	-18	-20	-26	51
389	-389	-154	-49	-55	-81	-19	-17	-20	-32	510
390	-771	-116	-46	-52	-676	-16	-16	-4	-4	213
391	-3260	31	54	48	-130	41	73	59	59	549
392	-333	-35	6	4	-11	2	11	-24	-68	192
393	-235	-90	-24	-28	-399	-16	-11	-6	-6	27
394	-702	-165	-64	-66	-951	-30	-30	-17	-17	15
395	-11045	-55	2	-4	-57	-9	-3	-22	-126	312
396	-557	-26	11	9	-164	-11	-4	-5	-11	33
397	-290	-163	-52	-52	-424	-18	-12	-15	-15	21
398	-2968	-23	0	-4	-257	3	1	4	4	60
399	-816	-63	-20	-21	-321	-16	-16	-20	-29	39
400	-813	17	24	24	-267	-6	8	35	35	150

KEY
K kdd8f002
L ok8f112
M ok8f311
N ok8f312
O AntAdaptive1
P uttno8fl
Q uttno8flf
R INQ610
S INQ613

TREC-8 Adaptive Filtering Results, LF1 Measure, Year 92 (Part 1)

Topic	A	B	C	D	E	F	G	H	I	J	Maximal
351	13	6	13	11	-17	-17	-2	-2	-266	-266	33
352	-8	-6	-8	-8	-2	-2	-2	-2	-28	-28	171
353	-1	-2	-1	-1	-1	-1	-2	-2	-227	-227	45
354	-4	4	-4	-4	0	0	-2	-4	-488	-488	66
355	-8	-8	-8	-8	-2	-2	-2	0	-38	-38	3
356	-20	-3	-16	-16	0	0	-2	-2	-52	-52	24
357	-5	-11	-1	-1	10	10	-2	-2	-119	-119	72
358	-18	-6	-16	-16	0	0	-2	-2	-74	-74	0
359	-12	-8	-8	-8	-83	-83	-2	-204	-77	-77	24
360	-8	-6	-8	-8	-6	-6	0	-2	-103	-103	6
361	-12	-18	-12	-12	-6	-6	-2	-2	-164	-164	0
362	-14	-12	-14	-14	-6	-6	-2	-2	-80	-80	0
363	-10	-1	-8	-8	-34	-34	-2	-8	-462	-460	6
364	-9	-6	-7	-7	-29	-29	3	1	-11	-11	3
365	18	11	18	14	22	22	4	6	-20	-20	24
366	-3	-3	0	0	4	4	-2	-2	-35	-35	12
367	-3	-10	-3	-3	5	5	0	-2	-130	-132	36
368	-10	-2	-8	-8	-2	-2	-2	-4	-10	-10	0
369	-9	-9	-9	-9	-2	-2	-2	-2	-101	-101	3
370	17	6	1	1	0	0	0	-3	-81	-83	66
371	-12	-6	-8	-8	-26	-26	-2	-18	-801	-801	3
372	-7	-18	-5	-5	-13	-13	-4	-2	-76	-76	9
373	-20	-14	-16	-16	-18	-18	0	-2	-6	-6	6
374	-5	-1	-5	-5	-5	-5	0	1	-102	-102	57
375	-9	18	-9	-9	-9	-7	3	-1	-69	-69	30
376	-5	-8	-5	-5	-63	-63	-2	-31	-978	-978	39
377	-8	1	-8	-8	-32	-32	0	-2	-46	-46	15
378	-14	-10	-10	-10	-6	-4	-2	-2	-828	-885	153
379	-18	-6	-14	-14	0	0	-2	0	-166	-166	0
380	-2	-12	-2	-2	-22	-22	-2	-2	-20	-20	0
381	-12	-8	-10	-10	-36	-36	-2	-2	-130	-130	15
382	-3	-7	-3	-5	-5	-5	3	-5	-117	-117	6
383	-13	-6	-11	-11	-50	-50	0	-138	-374	-376	45
384	-14	-8	-10	-10	-6	-6	-2	-4	-48	-48	3
385	-8	-10	-8	-8	0	0	-2	-10	-343	-345	27
386	-14	-10	-10	-10	-5	-5	0	-8	-167	-167	3
387	-9	-6	-9	-9	-23	-23	-2	-10	-63	-63	12
388	-4	-6	-4	-4	1	1	-2	-2	-54	-54	9
389	-8	-6	-6	-6	0	0	-2	-4	-108	-108	123
390	-3	-6	-1	1	-1	-1	0	-69	-231	-231	9
391	18	-6	27	29	5	9	2	-77	-1046	-1183	168
392	-4	2	-4	-4	3	5	13	7	-30	-30	96
393	-18	-10	-14	-14	-11	-11	-2	-2	-68	-68	12
394	-8	-8	-8	-8	-54	-54	-22	-8	-234	-234	0
395	2	-3	5	5	2	2	13	3	-933	-971	126
396	3	2	0	3	9	9	9	2	-172	-174	18
397	-8	-7	-6	-6	-61	-61	-2	-2	-90	-90	6
398	8	14	8	12	18	18	-2	-2	-1064	-1096	30
399	-5	-8	-5	-5	-2	-2	-2	-2	-271	-271	12
400	44	7	44	52	-16	-16	9	-4	16	14	102

KEY

A	CL99afL1b
B	CL99afL1a
C	CL99afL1c
D	CL99afL1d
E	pir9LF1
F	pir9LF1a
G	IOWAF992
H	IOWAF993
I	kdd8f003
J	kdd8f001

TREC-8 Adaptive Filtering Results, LF1 Measure, Year 92 (Part 2)

Topic	K	L	M	N	O	P	Q	R	S	Maximal
351	-303	-29	0	-2	-93	1	-1	-14	-21	33
352	-28	-12	5	3	-1002	-5	-6	-16	-19	171
353	-225	-37	-11	-11	-112	-4	-2	-20	-29	45
354	-504	-33	-4	-3	-29	0	4	0	0	66
355	-38	-10	0	-2	-136	-16	-16	-20	-24	3
356	-52	-11	3	3	-146	3	3	6	6	24
357	-119	-9	3	1	-84	-12	-12	-17	-26	72
358	-74	-54	-20	-26	-330	-20	-20	-2	-2	0
359	-75	-95	-39	-41	-608	-19	-19	0	0	24
360	-103	-82	-17	-21	-312	-12	-4	4	4	6
361	-164	-140	-56	-56	-718	-18	-18	-20	-32	0
362	-80	-98	-44	-44	-480	-18	-18	-20	-24	0
363	-468	-118	-52	-56	-358	-40	-40	-20	-32	6
364	-11	-25	-11	-13	-27	-14	-14	-15	-15	3
365	-28	-4	8	8	-972	16	16	20	20	24
366	-35	-14	8	6	-736	4	4	0	0	12
367	-132	-42	-26	-40	-46	-4	-4	-20	-25	36
368	-10	-18	-10	-12	-34	-10	-10	-4	-4	0
369	-101	-2	-2	-2	-2	-8	-8	-9	-9	3
370	-73	-15	11	9	-202	3	8	3	3	66
371	-803	-65	-28	-31	-380	-14	-14	-14	-14	3
372	-76	-48	-12	-16	-456	-11	-11	-2	-4	9
373	-6	-63	-17	-21	-285	-12	-12	-17	-29	6
374	-102	-42	-9	-11	-44	-2	2	21	21	57
375	-75	-21	-13	-8	-826	-11	-11	7	7	30
376	-983	-149	-50	-58	-408	-26	-26	5	5	39
377	-46	-60	-20	-28	-278	-13	-14	-14	-17	15
378	-963	-154	-52	-55	-620	-37	-33	-20	-32	153
379	-166	-26	-6	-6	-254	-12	-12	-4	-4	0
380	-20	-82	-26	-30	-534	-12	-12	-6	-6	0
381	-130	-74	-33	-37	-530	-26	-26	-12	-12	15
382	-123	-19	-5	-9	-531	-9	-9	-7	-7	6
383	-370	-95	-20	-23	-774	-15	-15	-6	-6	45
384	-48	-71	-16	-20	-93	-12	-12	-8	-8	3
385	-347	-58	-15	-17	-551	-11	-11	-17	-16	27
386	-165	-39	-11	-15	-573	-10	-10	-11	-11	3
387	-63	-68	-32	-34	-317	-11	-9	-17	-20	12
388	-54	-35	-14	-13	-67	-14	-14	-20	-26	9
389	-108	-108	-31	-37	-81	-13	-11	-20	-32	123
390	-231	-111	-37	-43	-71	-13	-11	1	1	9
391	-942	-37	4	-2	-130	25	31	55	55	168
392	-42	-19	2	0	-11	11	20	-2	-6	96
393	-68	-57	-19	-21	-377	-9	-7	-1	-1	12
394	-234	-124	-42	-46	-712	-26	-26	-10	-10	0
395	-6238	-48	6	0	-57	-11	-4	-22	-35	126
396	-166	-30	9	5	-164	-7	-7	10	10	18
397	-90	-114	-38	-40	-126	-7	-9	-6	-6	6
398	-1038	-25	1	-3	-257	4	2	3	3	30
399	-271	-56	-20	-21	-174	-18	-18	-14	-14	12
400	14	24	33	33	-267	23	33	25	25	102

KEY
K kdd8f002
L ok8f112
M ok8f311
N ok8f312
O AntAdaptive1
P uttno8fl
Q uttno8flf
R INQ610
S INQ613

TREC-8 Adaptive Filtering Results, LF1 Measure, Year 93 (Part 1)

Topic	A	B	C	D	E	F	G	H	I	J	Maximal
351	10	9	10	6	3	3	0	-3	-344	-344	30
352	1	-3	1	1	0	0	0	0	-28	-28	195
353	0	0	0	0	-1	-1	0	0	-325	-327	33
354	3	0	3	3	3	3	0	-6	-441	-441	108
355	0	0	0	0	-8	-8	0	-2	-42	-42	3
356	3	3	3	3	0	0	0	0	-62	-62	33
357	0	0	0	0	16	16	0	0	-98	-100	93
358	-2	0	-2	-2	-6	-6	0	0	-86	-86	6
359	-2	-2	-2	-2	-12	-12	0	-175	-95	-95	42
360	0	0	0	0	-3	-3	-4	0	-165	-165	9
361	-2	-4	-2	-2	-14	-14	0	-4	-170	-170	3
362	1	-2	1	1	2	2	0	0	-55	-55	12
363	0	-2	0	0	-38	-38	0	-16	-962	-966	3
364	0	0	0	0	0	0	0	-2	-6	-6	0
365	6	-4	6	6	6	6	-2	0	-9	-9	6
366	0	0	9	36	16	16	0	0	-49	-51	39
367	0	0	0	0	-1	-1	0	-2	-310	-310	39
368	-2	3	-2	-2	-2	-2	0	-2	-15	-15	3
369	0	0	0	0	0	0	0	0	-204	-204	0
370	-2	0	0	3	0	0	0	0	-126	-134	60
371	0	0	0	0	-60	-60	0	-8	-959	-961	3
372	0	0	0	0	0	0	-1	0	-57	-57	12
373	-2	3	-2	-2	-7	-7	3	0	-11	-11	12
374	0	0	0	0	-6	-6	0	-2	-98	-98	66
375	0	3	0	0	-1	1	-4	-14	-84	-86	12
376	0	0	0	0	-43	-43	0	-27	-932	-920	36
377	0	0	0	0	0	0	-2	0	-45	-45	12
378	0	-4	0	0	0	0	0	0	-722	-738	72
379	0	0	0	0	0	0	0	-2	-192	-192	0
380	0	0	0	0	-37	-37	0	-2	-39	-39	3
381	-2	-2	-2	-2	-60	-60	0	0	-200	-200	0
382	0	0	0	0	-2	-2	-4	-12	-168	-168	3
383	0	-4	0	1	-57	-57	-4	-137	-443	-509	111
384	-2	-2	-2	-2	-9	-9	0	0	-156	-158	6
385	0	0	0	0	0	0	0	0	-234	-238	27
386	-6	-2	-6	-6	-8	-8	0	-16	-186	-186	9
387	0	0	0	0	0	0	0	-48	-58	-58	3
388	3	3	3	3	0	0	0	0	-34	-34	6
389	0	0	0	0	-2	-2	0	0	-127	-127	234
390	0	0	0	0	-4	-4	0	-61	-272	-276	105
391	6	0	17	21	-4	-1	9	-110	-1116	-1327	210
392	0	1	0	6	-1	1	0	0	-40	-40	57
393	0	-4	0	0	0	0	0	0	-75	-75	15
394	0	-2	0	0	-80	-80	-8	-28	-239	-239	3
395	0	0	0	-12	-2	-2	-1	-9	-522	-638	90
396	0	0	0	0	-9	-7	0	-3	-210	-226	6
397	0	0	0	0	-9	-9	-2	-2	-90	-90	12
398	6	6	6	4	1	1	0	-10	-969	-1019	12
399	0	3	0	1	1	1	0	0	-247	-247	12
400	1	6	1	-6	0	0	0	-4	-381	-381	21

KEY
A CL99afL1b
B CL99afL1a
C CL99afL1c
D CL99afL1d
E pir9LF1
F pir9LF1a
G IOWAF992
H IOWAF993
I kdd8f003
J kdd8f001

TREC-8 Adaptive Filtering Results, LF1 Measure, Year 93 (Part 2)

Topic	K	L	M	N	O	P	Q	R	S	Maximal
351	-448	8	8	8	0	10	10	-10	-44	30
352	-28	7	4	4	0	0	-1	-3	-49	195
353	-319	-7	-2	-2	-125	6	8	0	0	33
354	-712	4	3	3	0	1	11	0	0	108
355	-42	-4	-2	-2	-158	-2	-2	0	-8	3
356	-62	-7	1	1	-96	3	1	6	6	33
357	-96	4	3	3	0	0	0	0	0	93
358	-86	-24	-10	-10	-214	-6	-6	-2	-2	6
359	-95	-27	-9	-9	-4	-6	-6	-4	-4	42
360	-165	-18	-4	-4	-274	0	0	-1	-1	9
361	-170	-32	-10	-10	-284	0	0	0	0	3
362	-55	-24	-8	-8	-208	2	2	0	-2	12
363	-1010	-36	-16	-14	0	-2	-2	0	0	3
364	-6	-6	-4	-6	-8	-2	-2	-2	-10	0
365	-16	4	4	4	-30	6	6	6	6	6
366	-49	21	21	21	-266	39	39	16	16	39
367	-322	-3	-3	-3	-12	2	3	0	-11	39
368	-15	-10	-2	-2	-26	-4	-4	-11	-11	3
369	-208	-2	0	0	0	-2	-2	-8	-16	0
370	-123	-1	-1	-1	0	3	1	0	0	60
371	-955	-37	-12	-16	-260	-2	-2	-6	-10	3
372	-57	-1	7	7	-518	3	3	2	2	12
373	-11	-9	-3	-5	0	1	1	0	0	12
374	-98	4	10	10	0	-5	1	13	13	66
375	-82	-2	-1	0	-176	4	4	4	4	12
376	-926	-11	-4	-2	-24	-1	-1	6	6	36
377	-44	3	3	3	1	1	-1	0	0	12
378	-698	-18	-4	-6	-382	-4	0	0	0	72
379	-192	-16	-4	-4	-184	-4	-4	-4	-4	0
380	-39	-27	-15	-15	-468	1	1	-14	-13	3
381	-200	-40	-16	-18	-310	-4	-4	-8	-20	0
382	-168	-18	-6	-8	-2	0	-2	-10	-16	3
383	-464	-9	-6	0	-13	4	4	-6	-8	111
384	-180	-30	-12	-12	0	-4	-4	-12	-14	6
385	-238	4	3	3	-2	3	3	0	-2	27
386	-184	-36	-6	-8	0	0	0	-6	-18	9
387	-58	-38	-20	-18	-4	-10	-12	0	-6	3
388	-34	-11	-3	-3	-2	-2	-2	0	0	6
389	-129	-28	-12	-12	0	-4	-4	0	0	234
390	-272	-9	-10	-10	-477	-6	-6	-4	-4	105
391	-1088	62	24	24	0	6	20	16	16	210
392	-192	-6	0	0	0	-6	-2	-14	-17	57
393	-81	-15	-1	-3	-12	-1	0	-1	-1	15
394	-239	-30	-20	-18	-231	-4	-4	-4	-4	3
395	-4213	2	1	1	0	4	9	-2	-49	90
396	-214	3	1	1	0	-7	-3	-17	-19	6
397	-90	-29	-10	-10	-124	-5	3	-3	-3	12
398	-947	4	1	1	0	1	1	3	3	12
399	-245	-9	-1	-1	-117	-1	-1	-6	-7	12
400	-387	-4	-6	-6	0	-18	-14	-1	-1	21

KEY
K kdd8f002
L ok8f112
M ok8f311
N ok8f312
O AntAdaptive1
P uttno8fl
Q uttno8flf
R INQ610
S INQ613

TREC-8 Adaptive Filtering Results, LF1 Measure, Year 94 (Part 1)

Topic	A	B	C	D	E	F	G	H	I	J	Maximal
351	5	7	5	4	0	0	0	2	-339	-347	21
352	0	0	0	0	-3	-3	0	0	-76	-76	375
353	0	0	0	0	11	14	0	0	-59	-59	54
354	0	0	0	0	-8	-8	0	-8	-106	-106	39
355	0	-6	0	0	-10	-10	0	0	-62	-62	0
356	0	0	0	0	0	0	0	0	-70	-70	12
357	0	0	3	1	14	13	0	1	-79	-79	90
358	-2	0	-2	-2	-2	-2	0	0	-96	-96	0
359	-2	-2	-2	-2	-2	-2	0	-130	-193	-197	48
360	0	-2	0	0	-2	-2	-2	-75	-194	-193	27
361	-2	-2	-2	-2	-16	-16	0	-4	-156	-156	3
362	0	0	0	0	0	0	0	0	-114	-116	3
363	0	0	0	0	-42	-42	0	-33	-1129	-1137	9
364	0	0	0	0	0	0	-2	0	2	2	6
365	3	0	3	3	3	3	0	-2	-18	-18	3
366	0	-2	1	-4	-17	-9	0	-4	-143	-143	12
367	0	0	0	9	5	5	-2	-4	-455	-461	51
368	0	0	0	0	6	6	0	-6	-50	-50	12
369	0	0	0	-6	-6	-6	0	0	-292	-292	0
370	0	0	-2	-2	0	0	-2	0	-177	-178	33
371	0	-2	0	0	-68	-68	0	-14	-1008	-1010	0
372	0	0	0	0	1	3	-1	0	-96	-96	24
373	3	3	3	3	-2	-2	9	0	-6	-6	24
374	0	0	0	0	-6	-6	0	-2	-150	-150	111
375	0	0	0	0	-5	-5	3	1	-97	-97	18
376	0	3	0	0	3	-15	0	-29	-140	-140	15
377	0	-2	0	0	3	3	0	0	-57	-59	18
378	0	0	0	0	0	0	0	0	-615	-626	60
379	-2	-2	-2	-2	0	0	0	0	-178	-178	0
380	0	0	0	0	-50	-50	0	0	-42	-44	0
381	0	0	0	0	-33	-33	0	0	-208	-208	6
382	4	0	4	4	7	9	3	1	-189	-191	9
383	0	0	0	-3	-24	-24	-2	-77	-503	-601	90
384	-2	0	-2	-2	-2	-2	0	0	-853	-857	3
385	0	3	0	0	6	6	0	0	-280	-290	54
386	0	0	0	0	0	0	-2	-16	-192	-192	9
387	0	0	0	2	2	-1	-2	-38	-48	-47	24
388	0	0	0	0	-1	-1	0	0	-17	-17	36
389	-2	-2	-2	-2	0	0	0	-2	-152	-152	153
390	0	3	3	3	3	3	3	-53	-274	-275	99
391	22	0	38	34	0	0	-14	-118	-1313	-1700	171
392	0	0	0	0	4	4	0	-22	-95	-95	39
393	0	0	0	0	0	0	0	0	-56	-56	0
394	3	0	3	3	-23	-23	-10	-30	-229	-229	12
395	0	0	0	-6	0	0	1	-10	-354	-465	96
396	0	0	0	0	1	1	0	-8	-171	-175	9
397	0	0	0	0	-4	-4	-2	-14	-110	-110	3
398	0	0	0	-2	0	0	0	-6	-997	-1031	18
399	0	0	0	3	3	3	0	-2	-292	-292	15
400	3	3	3	4	0	0	0	0	-432	-434	27

KEY

A	CL99afL1b
B	CL99afL1a
C	CL99afL1c
D	CL99afL1d
E	pir9LF1
F	pir9LF1a
G	IOWAF992
H	IOWAF993
I	kdd8f003
J	kdd8f001

TREC-8 Adaptive Filtering Results, LF1 Measure, Year 94 (Part 2)

Topic	K	L	M	N	O	P	Q	R	S	Maximal
351	-447	5	7	7	0	5	5	-12	-22	21
352	-76	16	12	12	0	-1	-4	-4	-2	375
353	-59	4	4	4	-16	16	18	0	0	54
354	-482	0	-4	-4	0	0	-6	0	0	39
355	-62	-6	-2	-2	-142	-2	-2	0	0	0
356	-70	-6	-4	-4	-80	-2	-2	0	0	12
357	-79	-3	0	0	0	2	2	0	0	90
358	-96	-10	-4	-4	-228	-2	-2	-2	-2	0
359	-195	-15	-8	-6	-10	-4	-4	-4	-4	48
360	-194	3	1	1	-292	-5	-13	0	0	27
361	-156	-14	-2	-2	0	-2	-2	0	0	3
362	-112	-26	-2	-2	0	0	0	0	0	3
363	-1151	-28	-8	-8	0	-2	-2	0	0	9
364	2	2	4	2	-126	3	3	0	-4	6
365	-21	-2	0	0	0	3	3	3	3	3
366	-149	8	8	8	0	0	2	0	0	12
367	-529	-2	-2	-4	-4	3	8	0	-12	51
368	-48	4	5	5	-44	7	4	-2	6	12
369	-294	-2	-2	-2	-4	-4	-4	0	-4	0
370	-159	0	-2	-2	0	-2	-6	0	0	33
371	-1002	-20	-8	-6	-362	-6	-6	0	-8	0
372	-96	10	11	11	0	4	4	7	7	24
373	-6	5	6	6	0	-4	-4	0	0	24
374	-122	25	29	29	0	22	21	14	14	111
375	-105	-7	0	0	0	-3	-5	-3	-3	18
376	-132	-2	-2	-2	-10	3	1	0	0	15
377	-65	1	3	3	1	3	3	0	-11	18
378	-571	0	0	0	0	0	0	0	0	60
379	-178	-20	-8	-10	-220	-6	-6	-8	-8	0
380	-60	-14	-2	-2	0	-2	-2	0	-10	0
381	-208	-1	-2	-2	-104	0	0	0	0	6
382	-195	5	7	9	0	7	9	0	-6	9
383	-500	-2	0	0	0	-2	-2	0	-6	90
384	-1381	-12	-2	-4	0	-2	-2	0	-10	3
385	-286	7	6	6	0	6	12	0	5	54
386	-190	-6	-4	-4	0	-2	-2	0	0	9
387	-52	-23	-1	1	-2	3	0	0	0	24
388	-17	18	18	18	0	-2	-2	0	0	36
389	-152	-18	-6	-6	0	-2	-2	0	0	153
390	-268	4	1	1	-128	3	1	-1	-1	99
391	-1230	6	26	26	0	10	22	-12	-12	171
392	-99	-10	4	4	0	-3	-7	-8	-45	39
393	-86	-18	-4	-4	-10	-6	-4	-4	-4	0
394	-229	-11	-2	-2	-8	0	0	-3	-3	12
395	-594	-9	-5	-5	0	-2	-8	2	-42	96
396	-177	1	1	3	0	3	6	2	-2	9
397	-110	-20	-4	-2	-174	-6	-6	-6	-6	3
398	-983	-2	-2	-2	0	-2	-2	-2	-2	18
399	-300	2	1	1	-30	3	3	0	-8	15
400	-440	-3	-3	-3	0	-11	-11	11	11	27

KEY
K kdd8f002
L ok8f112
M ok8f311
N ok8f312
O AntAdaptive1
P uttno8lfl
Q uttno8lflf
R INQ610
S INQ613

TREC-8 Adaptive Filtering Results, LF2 Measure, Years 92-94

Topic	A	B	C	D	E	F	G	H	I	J	K	L	M	Maximal
351	40	2	2	-1	17	28	27	34	35	9	18	12	-12	84
352	-2	-2	-2	-1	143	148	146	0	2	0	0	11	64	741
353	1	19	6	7	-11	3	3	30	27	0	0	-10	-13	132
354	1	-3	-3	-1	-3	9	5	39	26	19	16	0	0	213
355	-4	-19	-19	-1	-10	-3	-2	-10	-10	-10	-12	-10	-16	6
356	-5	-2	-2	-1	-9	3	3	4	5	0	0	12	12	69
357	3	37	63	-1	11	13	11	-2	-2	8	11	-7	-10	255
358	-10	-20	-20	-1	-44	-20	-17	-14	-14	-5	-9	-3	-3	6
359	-6	-108	-108	-1	-61	-25	-25	-13	-13	-10	-17	-4	-4	114
360	-4	1	-3	-5	-40	-4	-4	-1	-3	1	1	9	9	42
361	-8	-103	-103	-1	-93	-34	-34	-10	-10	0	0	-10	-16	6
362	-5	-6	-6	-1	-71	-24	-24	-5	-5	6	6	-10	-10	15
363	-4	-56	-56	-6	-91	-39	-38	-22	-22	0	0	-10	-16	18
364	-2	-32	-32	2	-10	-4	-1	-5	-5	-2	-5	-7	-13	9
365	26	32	32	6	10	18	18	29	29	6	6	31	31	33
366	45	17	10	-1	34	36	47	54	53	5	7	23	23	63
367	9	-1	-2	0	-17	-19	-11	19	17	0	0	-10	-15	126
368	-5	0	0	-1	-6	0	1	-2	1	-1	-1	-7	3	15
369	-6	-5	-5	-1	-3	-2	-2	-7	-7	-3	-4	-7	-13	3
370	6	0	0	-4	-1	14	12	9	5	0	0	3	3	159
371	-4	-64	-64	-1	-58	-25	-24	-11	-11	0	0	-10	-16	6
372	-1	-3	-4	-1	-5	10	17	4	4	3	3	17	16	45
373	-6	-10	-10	14	-24	-1	2	-6	-6	0	0	-7	-13	42
374	-1	6	6	6	28	37	54	60	44	0	0	51	51	234
375	-3	5	-2	0	-6	-1	-1	2	2	5	-5	13	13	60
376	-1	-67	-67	-10	-75	-28	-25	-10	-9	0	0	13	13	90
377	-4	1	0	8	-16	-3	1	2	2	20	22	-4	-5	45
378	-5	-8	-9	-2	-80	-26	-23	-15	-17	-7	-18	-10	-16	285
379	-8	-12	-12	-1	-31	-10	-9	-11	-11	0	0	-8	-8	0
380	-1	-133	-133	-1	-60	-22	-20	-5	-5	-7	-15	-10	-13	3
381	-6	-22	-22	-1	-53	-27	-24	-15	-15	-13	-19	-10	-16	21
382	4	3	2	3	-10	1	4	5	5	5	-3	-7	-13	18
383	-2	-51	-53	-1	-44	-8	-7	-2	-2	6	7	-3	-7	246
384	-7	-20	-20	-1	-55	-18	-15	-9	-9	0	0	-10	-16	12
385	-4	17	13	-1	-11	5	5	11	5	0	0	-7	13	108
386	-8	-18	-18	-1	-39	-12	-9	-6	-6	-3	-4	-7	-13	21
387	0	-25	-25	-6	-57	-22	-19	-7	-5	-1	-6	-7	-10	39
388	4	6	13	-1	0	17	13	-9	-9	-1	7	-10	-10	51
389	-4	4	2	-1	-74	-26	-23	-7	-8	0	0	-10	-16	510
390	4	-3	-3	-2	-49	-23	-20	-5	-5	0	0	1	1	213
391	118	24	20	37	96	106	72	110	84	77	68	154	154	549
392	4	15	22	29	31	39	31	44	38	0	0	12	-4	192
393	-7	-27	-27	-1	-42	-11	-9	-4	-5	0	0	0	0	27
394	-1	-129	-129	-7	-81	-33	-32	-15	-15	0	0	-7	-7	15
395	-1	4	7	20	-10	5	25	34	31	0	0	13	12	312
396	8	13	15	8	-1	13	15	10	5	14	11	11	8	33
397	-3	-85	-85	-5	-80	-26	-26	0	-6	-1	-1	-6	-6	21
398	20	13	11	-1	-1	8	13	11	12	3	3	5	5	60
399	4	1	1	1	-24	-6	-7	-5	-5	0	0	-10	-13	39
400	64	1	1	11	47	48	59	52	48	0	0	43	43	150

KEY
A CL99afL2
B pir9LF2
C pir9LF2a
D IOWAF991
E ok8f122
F ok8f222
G ok8f321
H uttno8lf2f
I uttno8lf2
J uttno8lf1p
K uttno8lf2p
L INQ612
M INQ611

TREC-8 Adaptive Filtering Results, LF2 Measure, Year 92

Topic	A	B	C	D	E	F	G	H	I	J	K	L	M	Maximal
351	18	-9	-9	-1	-4	9	9	10	11	-2	3	5	3	33
352	-4	-4	-4	-1	27	31	44	1	2	0	0	10	10	171
353	1	0	0	-1	-17	-4	-4	2	1	0	0	-10	-13	45
354	-2	-4	-4	-1	-6	8	4	17	16	9	5	0	0	66
355	-4	-5	-5	-1	-5	-1	0	-8	-8	-3	-3	-10	-12	3
356	-8	0	0	-1	-4	3	3	3	3	0	0	6	6	24
357	1	4	16	-1	0	5	6	-6	-6	-1	-1	-7	-10	72
358	-8	-7	-7	-1	-27	-13	-10	-10	-10	-4	-6	-1	-1	0
359	-4	-97	-97	0	-43	-19	-18	-8	-8	-10	-16	0	0	24
360	-4	0	0	-3	-38	-9	-7	1	-3	0	0	5	5	6
361	-6	-28	-28	-1	-70	-28	-28	-9	-9	0	0	-10	-16	0
362	-7	-6	-6	-1	-49	-22	-22	-9	-9	0	0	-10	-12	0
363	-4	-51	-51	-3	-59	-28	-26	-20	-20	0	0	-10	-16	6
364	-2	-32	-32	2	-11	-5	-4	-7	-7	-2	-4	-6	-6	3
365	18	23	23	7	7	16	13	20	20	6	6	22	22	24
366	6	2	2	0	-1	9	10	8	8	-1	-1	3	3	12
367	0	-8	-8	-1	-12	-17	-10	4	4	0	0	-10	-8	36
368	-4	-3	-3	-1	-9	-6	-5	-5	-5	0	0	-2	-2	0
369	-3	-1	-1	-1	-1	-1	-1	-4	-4	-1	-1	-3	-3	3
370	2	0	0	0	3	12	13	10	3	0	0	3	3	66
371	-4	-33	-33	-1	-31	-14	-14	-7	-7	0	0	-7	-7	3
372	-1	-11	-11	-1	-21	-5	-3	-4	-4	0	0	2	1	9
373	-8	-9	-9	0	-30	-9	-7	-6	-6	0	0	-7	-13	6
374	-1	5	5	5	0	10	9	24	17	0	0	21	21	57
375	-3	6	3	2	0	1	-2	-1	-1	8	5	8	8	30
376	-1	-53	-53	-1	-73	-29	-25	-13	-13	0	0	7	7	39
377	-4	-2	-2	5	-24	-9	-5	-2	-3	10	9	-4	-4	15
378	-5	-7	-9	-1	-71	-23	-21	-10	-15	-6	-16	-10	-16	153
379	-7	-4	-4	-1	-13	-3	-3	-6	-6	0	0	-2	-2	0
380	-1	-33	-33	-1	-41	-15	-13	-6	-6	-7	-13	-3	-3	0
381	-5	-18	-18	0	-34	-17	-15	-13	-13	-10	-14	-6	-6	15
382	-1	-3	-3	1	-8	-3	-1	-3	-3	0	-7	-2	-2	6
383	-4	-32	-32	0	-43	-7	-7	-6	-6	4	4	-3	-3	45
384	-5	-11	-11	0	-34	-10	-8	-6	-6	0	0	-4	-4	3
385	-4	-1	-1	-1	-23	-4	-3	-4	-4	0	0	-7	-5	27
386	-5	-12	-12	-1	-18	-6	-4	-5	-5	-3	-3	-4	-4	3
387	-3	-21	-21	-3	-31	-14	-13	-3	-4	-4	-6	-7	-7	12
388	-2	1	1	-1	-16	-5	-7	-7	-7	-6	-10	-10	-10	9
389	-3	8	8	-1	-51	-17	-14	-4	-5	0	0	-10	-16	123
390	1	-1	-1	0	-54	-20	-17	-4	-5	0	0	2	2	9
391	34	20	16	26	6	17	20	41	38	33	31	68	68	168
392	1	9	20	20	26	35	26	37	28	0	0	14	15	96
393	-7	-24	-24	-1	-27	-9	-8	-2	-3	0	0	1	1	12
394	-4	-104	-104	-6	-62	-23	-21	-13	-13	0	0	-5	-5	0
395	9	6	9	21	-8	8	17	15	22	0	0	10	11	126
396	9	12	12	9	-6	10	12	4	4	14	14	14	14	18
397	-3	-78	-78	-1	-57	-20	-19	-3	-2	0	0	-3	-3	6
398	19	16	16	-1	-3	7	10	10	11	0	0	3	3	30
399	-1	-1	-1	-1	-25	-9	-10	-9	-9	-3	-3	-7	-7	12
400	56	1	1	5	49	49	45	50	52	0	0	29	29	102

KEY
A CL99afL2
B pir9LF2
C pir9LF2a
D IOWAF991
E ok8f122
F ok8f222
G ok8f321
H uttno8lf2f
I uttno8lf2
J uttno8lf1p
K uttno8lf2p
L INQ612
M INQ611

TREC-8 Adaptive Filtering Results, LF2 Measure, Year 93

Topic	A	B	C	D	E	F	G	H	I	J	K	L	M	Maximal
351	14	6	6	0	17	15	10	17	17	8	9	7	-10	30
352	2	0	0	0	37	37	32	1	1	0	0	3	1	195
353	0	4	4	2	1	2	2	10	9	0	0	0	0	33
354	3	1	1	0	3	4	3	19	17	15	16	0	0	108
355	0	-6	-6	0	-2	-1	-1	-1	-1	-4	-5	0	-4	3
356	3	0	0	0	-2	2	2	2	3	0	0	6	6	33
357	0	8	21	0	5	3	3	0	0	0	0	0	0	93
358	-1	-8	-8	0	-12	-5	-5	-3	-3	-1	-3	-1	-1	6
359	-1	-10	-10	0	-12	-3	-3	-3	-3	0	1	-2	-2	42
360	0	3	0	-1	-6	1	1	3	1	2	2	1	1	9
361	-1	-39	-39	0	-16	-5	-5	0	0	0	0	0	0	3
362	2	0	0	0	-9	-1	-1	4	4	6	6	0	2	12
363	0	-5	-5	-2	-18	-7	-8	-1	-1	0	0	0	0	3
364	0	0	0	0	-3	-3	-2	-1	-1	0	-1	-1	-5	0
365	5	6	6	-1	4	3	5	6	6	0	0	6	6	6
366	36	20	14	0	29	25	29	39	39	6	6	17	17	39
367	0	0	0	-1	-3	0	0	6	7	0	0	0	-1	39
368	-1	1	1	0	-5	-1	-1	-2	-2	0	0	-4	-4	3
369	0	-1	-1	0	-1	0	0	-1	-1	-1	-1	-4	-8	0
370	3	0	0	-2	0	8	5	2	3	0	0	0	0	60
371	0	-29	-29	0	-17	-8	-6	-1	-1	0	0	-3	-5	3
372	0	4	6	0	4	7	7	3	3	0	0	4	4	12
373	-1	0	0	2	-3	-1	0	2	2	0	0	0	0	12
374	0	3	3	1	7	6	14	13	0	0	0	14	14	66
375	0	0	-1	-3	1	3	1	5	6	-2	-4	5	5	12
376	0	-17	-17	-7	-1	2	1	1	1	0	0	6	6	36
377	0	2	2	0	6	4	3	1	2	5	5	0	3	12
378	0	-1	0	-1	9	-3	-2	-5	-2	-1	-2	0	0	72
379	0	-3	-3	0	-8	-2	-2	-2	-2	0	0	-2	-2	0
380	0	-47	-47	0	-12	-6	-6	2	2	1	-1	-7	-5	3
381	-1	-4	-4	-1	-20	-9	-8	-2	-2	-3	-5	-4	-10	0
382	0	-2	-2	-2	-9	-4	-3	-1	0	0	-1	-5	-8	3
383	2	-18	-19	-3	0	-2	0	5	5	3	1	0	-1	111
384	-1	-8	-8	0	-15	-6	-6	-2	-2	0	0	-6	-7	6
385	0	5	3	0	5	5	2	3	3	0	0	0	5	27
386	-3	-5	-5	0	-18	-4	-3	0	0	0	0	-3	-9	9
387	0	-2	-2	-2	-19	-9	-10	-7	-7	-1	-2	0	-3	3
388	3	2	1	0	-4	0	0	-1	-1	3	1	0	0	6
389	0	-3	-5	0	-14	-6	-6	-2	-2	0	0	0	0	234
390	0	-4	-4	-2	0	-5	-5	-3	-3	0	0	-2	-2	105
391	38	5	6	3	68	67	24	34	21	21	19	53	53	210
392	4	5	0	4	9	8	8	11	12	0	0	-1	-1	57
393	0	-1	-1	0	-6	0	1	0	1	0	0	1	1	15
394	0	-28	-28	-1	-15	-9	-10	-2	-2	0	0	-2	-2	3
395	-7	-1	-2	1	4	4	8	13	8	0	0	-1	-2	90
396	-1	-1	-1	-1	0	-2	1	0	-2	0	-2	-7	-8	6
397	0	-7	-7	-2	-13	-5	-5	6	-1	-1	-1	0	0	12
398	2	-1	-3	0	3	2	4	2	2	3	3	3	3	12
399	2	2	2	3	-3	1	1	1	1	3	3	-3	-2	12
400	2	0	0	3	-7	-6	5	-1	-3	0	0	1	1	21

KEY
A CL99afL2
B pir9LF2
C pir9LF2a
D IOWAF991
E ok8f122
F ok8f222
G ok8f321
H uttno8lf2f
I uttno8lf2
J uttno8lf1p
K uttno8lf2p
L INQ612
M INQ611

TREC-8 Adaptive Filtering Results, LF2 Measure, Year 94

Topic	A	B	C	D	E	F	G	H	I	J	K	L	M	Maximal
351	8	5	5	0	4	4	8	7	7	3	6	0	-5	21
352	0	2	2	0	79	80	70	-2	-1	0	0	-2	53	375
353	0	15	2	6	5	5	5	18	17	0	0	0	0	54
354	0	0	0	0	0	-3	-2	3	-7	-5	-5	0	0	39
355	0	-8	-8	0	-3	-1	-1	-1	-1	-3	-4	0	0	0
356	0	-2	-2	0	-3	-2	-2	-1	-1	0	0	0	0	12
357	2	25	26	0	6	5	2	4	4	9	12	0	0	90
358	-1	-5	-5	0	-5	-2	-2	-1	-1	0	0	-1	-1	0
359	-1	-1	-1	-1	-6	-3	-4	-2	-2	0	-2	-2	-2	48
360	0	-2	-3	-1	4	4	2	-5	-1	-1	-1	3	3	27
361	-1	-36	-36	0	-7	-1	-1	-1	-1	0	0	0	0	3
362	0	0	0	0	-13	-1	-1	0	0	0	0	0	0	3
363	0	0	0	-1	-14	-4	-4	-1	-1	0	0	0	0	9
364	0	0	0	0	4	4	5	3	3	0	0	0	-2	6
365	3	3	3	0	-1	-1	0	3	3	0	0	3	3	3
366	3	-5	-6	-1	6	2	8	7	6	0	2	3	3	12
367	9	7	6	2	-2	-2	-1	9	6	0	0	0	-6	51
368	0	2	2	0	8	7	7	5	8	-1	-1	-1	9	12
369	-3	-3	-3	0	-1	-1	-1	-2	-2	-1	-2	0	-2	0
370	1	0	0	-2	-4	-6	-6	-3	-1	0	0	0	0	33
371	0	-2	-2	0	-10	-3	-4	-3	-3	0	0	0	-4	0
372	0	4	1	0	12	8	13	5	5	3	3	11	11	24
373	3	-1	-1	12	9	9	9	-2	-2	0	0	0	0	24
374	0	-2	-2	0	21	21	31	23	27	0	0	16	16	111
375	0	-1	-4	1	-7	-5	0	-2	-3	-1	-6	0	0	18
376	0	3	3	-2	-1	-1	-1	2	3	0	0	0	0	15
377	0	1	0	3	2	2	3	3	3	5	8	0	-4	18
378	0	0	0	0	0	0	0	0	0	0	0	0	0	60
379	-1	-5	-5	0	-10	-5	-4	-3	-3	0	0	-4	-4	0
380	0	-53	-53	0	-7	-1	-1	-1	-1	-1	-1	0	-5	0
381	0	0	0	0	1	-1	-1	0	0	0	0	0	0	6
382	5	8	7	4	7	8	8	9	8	5	5	0	-3	9
383	0	-1	-2	2	-1	1	0	-1	-1	-1	2	0	-3	90
384	-1	-1	-1	-1	-6	-2	-1	-1	-1	0	0	0	-5	3
385	0	13	11	0	7	4	6	12	6	0	0	0	13	54
386	0	-1	-1	0	-3	-2	-2	-1	-1	0	-1	0	0	9
387	3	-2	-2	-1	-7	1	4	3	6	4	2	0	0	24
388	3	3	11	0	20	22	20	-1	-1	2	16	0	0	36
389	-1	-1	-1	0	-9	-3	-3	-1	-1	0	0	0	0	153
390	3	2	2	0	5	2	2	2	3	0	0	1	1	99
391	46	-1	-2	8	22	22	28	35	25	23	18	33	33	171
392	-1	1	2	5	-4	-4	-3	-4	-2	0	0	-1	-18	39
393	0	-2	-2	0	-9	-2	-2	-2	-3	0	0	-2	-2	0
394	3	3	3	0	-4	-1	-1	0	0	0	0	0	0	12
395	-3	-1	0	-2	-6	-7	0	6	1	0	0	4	3	96
396	0	2	4	0	5	5	2	6	3	0	-1	4	2	9
397	0	0	0	-2	-10	-1	-2	-3	-3	0	0	-3	-3	3
398	-1	-2	-2	0	-1	-1	-1	-1	-1	0	0	-1	-1	18
399	3	0	0	-1	4	2	2	3	3	0	0	0	-4	15
400	6	0	0	3	5	5	9	3	-1	0	0	13	13	27

KEY

A	CL99afL2
B	pir9LF2
C	pir9LF2a
D	IOWAF991
E	ok8f122
F	ok8f222
G	ok8f321
H	uttno8lf2f
I	uttno8lf2
J	uttno8lf1p
K	uttno8lf2p
L	INQ612
M	INQ611

TREC-8 Batch Filtering Results, LF1 Measure

Topic	plt8f1	plt8f2	CL99bFL1	pirc9BF1	Mer8BaLF1	AntBatch1	Scai8Ft	Maximal
351	22	25	20	25	-320	15	0	51
352	82	115	12	130	100	9	149	570
353	32	42	4	28	21	-1001	0	87
354	-9	-9	-2	-7	-15	-261	0	147
355	0	0	0	0	0	-324	0	3
356	7	7	-5	4	-345	0	-2	45
357	47	41	34	59	46	-43	3	183
358	0	0	0	0	0	0	0	6
359	-5	-5	0	0	0	-992	-8	90
360	0	0	0	0	3	-1002	0	36
361	0	0	-6	0	0	0	0	6
362	0	0	1	0	0	0	-2	15
363	0	0	-2	-4	-4	0	0	12
364	0	0	0	0	-2	-496	0	6
365	-16	-28	-2	4	0	3	0	9
366	0	0	4	3	22	-1002	0	51
367	0	5	0	4	4	-28	0	90
368	0	0	0	0	0	-55	0	15
369	0	0	0	0	0	0	0	0
370	-20	-20	-2	-13	-4	-389	0	93
371	0	0	0	0	0	0	0	3
372	3	3	-6	1	0	0	-4	36
373	0	0	13	0	0	0	0	36
374	5	5	-4	-19	12	-152	-2	177
375	-2	1	1	3	3	3	0	30
376	3	3	-9	0	0	-366	0	51
377	6	6	9	2	3	0	0	30
378	-25	-14	-12	-12	-11	-11	5	132
379	0	0	-8	0	0	0	0	0
380	0	0	0	0	0	0	0	3
381	0	0	-6	0	0	0	0	6
382	-2	-2	7	-2	0	-1002	0	12
383	97	113	-2	72	27	-7	-2	201
384	0	0	0	0	0	0	0	9
385	9	14	24	28	19	11	0	81
386	0	0	0	-2	0	0	0	18
387	-10	-10	-6	-1	3	-1002	0	27
388	0	0	-3	0	0	-323	0	42
389	145	113	122	76	31	176	218	387
390	-4	-4	0	0	3	-2	-4	204
391	36	-18	57	-35	-16	-22	-138	381
392	-32	-32	-5	2	-5	-21	4	96
393	7	7	-2	1	3	-14	0	15
394	0	0	-8	0	0	0	0	15
395	-8	-1	-6	-50	-1	-115	-9	186
396	1	1	-6	4	0	0	0	15
397	-2	-2	0	0	-6	0	0	15
398	9	9	-1	0	4	-3	0	30
399	0	0	4	3	0	0	0	27
400	-13	-8	3	-9	5	-20	0	48

TREC-8 Batch Filtering Results, LF2 Measure

Topic	CL99bfl2	pir9BF2	Mer8BaLF2	Maximal
351	24	28	-136	51
352	56	238	185	570
353	5	44	41	87
354	-1	6	0	147
355	0	0	0	3
356	-1	6	-6	45
357	50	69	56	183
358	0	0	0	6
359	0	-1	0	90
360	0	3	3	36
361	-3	0	0	6
362	2	0	0	15
363	-1	-2	-2	12
364	0	0	-1	6
365	-1	5	0	9
366	6	6	23	51
367	0	13	9	90
368	0	0	0	15
369	0	0	0	0
370	-1	-2	-9	93
371	0	0	0	3
372	-3	8	0	36
373	14	0	0	36
374	-3	6	42	177
375	2	6	5	30
376	-3	0	0	51
377	11	6	3	30
378	-6	-12	-19	132
379	-4	0	0	0
380	0	0	0	3
381	-3	0	0	6
382	8	2	0	12
383	0	103	59	201
384	0	0	0	9
385	26	35	31	81
386	0	-1	0	18
387	-3	0	3	27
388	0	1	0	42
389	213	203	32	387
390	0	0	3	204
391	59	39	68	381
392	2	23	15	96
393	1	5	3	15
394	-4	0	0	15
395	-3	16	44	186
396	-3	2	0	15
397	0	0	-3	15
398	2	4	5	30
399	4	5	0	27
400	5	11	9	48

TREC-8 Routing Filtering Results, Average Precision

Topic	plt8r2	plt8r1	pirc9R1	pirc9R2	dso99rt2	dso99rt1	S2N2	Mer8R1	Mer8R2	umrlsi	umrqz
351	0.664	0.652	0.470	0.586	0.688	0.691	0.589	0.747	0.030	0.524	0.525
352	0.466	0.432	0.533	0.460	0.564	0.565	0.512	0.465	0.201	0.534	0.533
353	0.626	0.495	0.841	0.854	0.805	0.765	0.626	0.641	0.048	0.861	0.861
354	0.170	0.203	0.401	0.191	0.301	0.325	0.069	0.095	0.000	0.004	0.298
355	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.002	0.004	1.000	1.000
356	0.187	0.207	0.380	0.117	0.218	0.096	0.137	0.099	0.111	0.161	0.161
357	0.498	0.437	0.563	0.515	0.647	0.639	0.397	0.527	0.000	0.624	0.635
358	0.000	0.000	0.106	0.150	0.004	0.000	0.001	0.000	0.000	0.002	0.001
359	0.066	0.066	0.043	0.035	0.124	0.160	0.118	0.067	0.000	0.088	0.089
360	0.314	0.346	0.070	0.061	0.265	0.227	0.154	0.166	0.336	0.000	0.504
361	0.000	0.000	0.000	0.000	0.010	0.003	0.015	0.001	0.001	0.012	0.012
362	0.242	0.301	0.382	0.424	0.596	0.628	0.034	0.292	0.292	0.516	0.514
363	0.000	0.000	0.364	0.043	0.073	0.078	0.112	0.000	0.007	0.052	0.050
364	0.095	0.191	0.667	0.700	0.583	0.583	0.625	0.000	0.029	0.750	0.750
365	0.041	0.092	1.000	1.000	1.000	1.000	0.700	0.656	0.226	0.756	0.756
366	0.373	0.424	0.831	0.810	0.855	0.845	0.922	0.712	0.087	0.866	0.870
367	0.236	0.099	0.514	0.361	0.521	0.461	0.098	0.254	0.019	0.077	0.067
368	0.278	0.301	0.754	0.778	0.853	0.786	0.268	0.335	0.335	0.498	0.498
369	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
370	0.201	0.191	0.239	0.208	0.294	0.327	0.078	0.114	0.043	0.096	0.242
371	0.000	0.001	0.005	0.003	0.024	0.020	0.002	0.000	0.000	0.015	0.015
372	0.634	0.648	0.784	0.769	0.806	0.824	0.648	0.680	0.006	0.673	0.673
373	0.315	0.283	0.601	0.213	0.422	0.479	0.245	0.006	0.000	0.589	0.588
374	0.182	0.337	0.511	0.513	0.490	0.542	0.512	0.403	0.000	0.500	0.503
375	0.042	0.123	0.607	0.583	0.420	0.376	0.524	0.314	0.000	0.146	0.277
376	0.121	0.104	0.100	0.095	0.341	0.362	0.056	0.179	0.000	0.061	0.035
377	0.205	0.260	0.606	0.585	0.797	0.644	0.796	0.595	0.475	0.591	0.607
378	0.156	0.113	0.145	0.122	0.225	0.220	0.144	0.110	0.025	0.173	0.170
379	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
380	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
381	0.007	0.009	0.020	0.041	0.016	0.019	0.036	0.026	0.000	0.037	0.019
382	0.074	0.072	0.671	0.710	0.689	0.755	0.532	0.028	0.011	0.575	0.608
383	0.536	0.568	0.618	0.221	0.639	0.625	0.321	0.409	0.000	0.286	0.478
384	0.185	0.132	0.083	0.376	0.105	0.108	0.036	0.001	0.002	0.030	0.031
385	0.546	0.627	0.661	0.508	0.762	0.760	0.422	0.516	0.173	0.550	0.571
386	0.086	0.063	0.090	0.044	0.083	0.029	0.084	0.064	0.059	0.374	0.373
387	0.535	0.612	0.165	0.115	0.492	0.497	0.305	0.126	0.103	0.355	0.352
388	0.376	0.422	0.598	0.543	0.506	0.501	0.127	0.247	0.010	0.595	0.594
389	0.283	0.390	0.856	0.787	0.865	0.844	0.377	0.785	0.000	0.840	0.849
390	0.564	0.540	0.687	0.592	0.790	0.746	0.106	0.061	0.104	0.669	0.704
391	0.308	0.312	0.288	0.302	0.432	0.469	0.407	0.455	0.255	0.245	0.285
392	0.218	0.274	0.373	0.375	0.389	0.393	0.373	0.253	0.000	0.379	0.381
393	0.352	0.228	0.598	0.241	0.502	0.488	0.045	0.601	0.050	0.215	0.236
394	0.036	0.079	0.066	0.055	0.061	0.011	0.051	0.032	0.032	0.000	0.015
395	0.324	0.282	0.278	0.318	0.407	0.357	0.252	0.380	0.205	0.002	0.169
396	0.528	0.276	0.044	0.419	0.540	0.616	0.557	0.239	0.211	0.276	0.276
397	0.390	0.260	0.439	0.830	0.440	0.399	0.180	0.122	0.082	0.163	0.620
398	0.412	0.264	0.547	0.465	0.668	0.647	0.353	0.276	0.378	0.554	0.576
399	0.103	0.168	0.320	0.251	0.233	0.181	0.098	0.007	0.016	0.232	0.239
400	0.419	0.404	0.662	0.580	0.535	0.460	0.325	0.450	0.411	0.667	0.665
Mean	0.288	0.286	0.432	0.399	0.462	0.451	0.307	0.271	0.108	0.364	0.406

TREC - 8 Interactive Track

Topics & assessor-determined instances

Number:

408i

Title:

tropical storms

Description:

What tropical storms (hurricanes and typhoons) have caused property damage and/or loss of life?

Instances:

In the time allotted, please find as many DIFFERENT storms of the sort described above as you can. Please save at least one document for EACH such DIFFERENT storm.

If one document discusses several such storms, then you need not save other documents that repeat those, since your goal is to identify as many DIFFERENT storms of the sort described above as possible.

Instance#

Instance gloss

- 1 Bangladesh cyclone Apr/May 91
- 2 Hurricane Huao 1989
- 3 typhoon 19
- 4 Hurricane Gilbert 1988
- 5 Typhoon Mireille 1991
- 6 Tropical storm Ted 1992
- 7 Hurricane Andrew 1992
- 8 Hurricane Iniki hit Kauau 1992
- 9 Hurricane Betsy 1965
- 10 Hurricane Calvin 1963
- 11 Hurricane Lydia 1993
- 12 Typhoon Vernon 1993
- 13 Tropical storm Cindy 1993
- 14 Tropical storm Bret
- 15 Cyclone Ofa 1992
- 16 Cyclone Val 1993
- 17 Bangladesh cyclone 1994
- 18 Tropical storm Debbie 1994
- 19 Typhoon Tim 1994
- 20 Tropical storm Alberto 1994
- 21 Typhoon Orchid 1994
- 22 Typhoon Fred 1994
- 23 Typhoon Doug 1994
- 24 Typhoon Seth 1994

Number:

414i

Title:

Cuba, sugar, imports

Description:

What countries import Cuban sugar?

Instances:

In the time allotted, please find as many DIFFERENT countries of the sort described above as you can. Please save at least one document for EACH such DIFFERENT country.

If one document discusses several such countries, then you need not save other documents that repeat those, since your goal is to identify as many DIFFERENT countries of the sort described

above as possible.

Instance#	Instance gloss
1	Soviet Union, Russia
2	China
3	Japan
4	Canada
5	Kazakhstan
6	Latvia
7	Iran
8	South Korea
9	Portugal
10	Mexico
11	Indonesia
12	Italy

Number:
428i

Title:
declining birth rates

Description:
What countries other than the US and China have or have had
a declining birth rate?

Instances:
In the time allotted, please find as many DIFFERENT countries of
the sort described above as you can. Please save at least one
document for EACH such DIFFERENT country.
If one document discusses several such countries, then you need
not save other documents that repeat those, since your goal
is to identify as many DIFFERENT countries of the sort described
above as possible.

Instance#	Instance gloss
1	Scotland
2	Greece
3	Italy
4	Japan
5	Germany
6	Belgium
7	France
8	UK (not a country but near enough)
9	India
10	Ireland
11	Russia
12	Thailand
13	Sri Lanka
14	Bulgaria
15	Spain
16	Poland
17	Egypt
18	Tunisia
19	Turkey
20	Singapore
21	Iran
22	Brazil
23	Tanzania

- 24 Namibia
- 25 South Africa
- 26 East Germany (in 1972 when separate country)

Number:
431i

Title:
robotic technology

Description:
What are the latest developments in robotic technology
and in its use?

Instances:
In the time allotted, please find as many DIFFERENT developments of
the sort described above as you can. Please save at least one
document for EACH such DIFFERENT development.
If one document discusses several such developments, then you need
not save other documents that repeat those, since your goal
is to identify as many DIFFERENT developments of the sort described
above as possible.

```

Instance#
|
| Instance gloss
| |
1 "clean room" applications for robots in health care (food)(pharmaceutical)
2 precision engineering
3 material handling and packaging (chemical) (food)
4 welding robotics
5 glass making
6 palletizing
7 assembling garments
8 robots unload trucks - moves products around in factories
9 robots make robots
10 robot can grasp soft/fragile items like a human hand
11 robotic storage devices and inventory stacking
12 electronic (computer, circuit board) machining, assembly, testing
13 robotic eyes for recognition systems, security, automation
14 robot make, screen and test materials (chemical)(pharmaceuticals)
15 robot telephone operators - speech interactive - office work
16 robotic cranes - remote
17 robot is dual ????? and diagnostic imaging system
18 robot cutting system (water jet)
19 robot traders
20 robotic engine assembly
21 industrialized car assembly and stamping
22 painting, spraying robotics
23 robots helping disabled people
24 deep sea robots / underwater crawlers
25 robotic vision systems
26 wall climbing robots
27 robotic rubber tree tappers
28 robotic vehicles repair oil rig damage underwater
29 medical robots help with human surgery
30 robotic nose (detects odors) can detect diseases in cows, contamination
31 robotic arm (space work)
32 robotic system cuts, cleans, washes, dries vehicle panels
33 robotic system helps genetic research
34 robotic trashmen (rubbish collectors)
35 robotic healthcare - hospital work - nursing - moving patients
36 robotic maintenance/repair work in hazardous area (nuclear)
37 robots for office - mail handling - selflearning
38 humanoid robot interacts with humans, takes care of self - selfexamines
39 robomom - controls time children watch TV

```


Number:

438i

Title:

tourism, increase

Description:

What countries have experienced an increase in tourism?

Instances:

In the time allotted, please find as many DIFFERENT countries of the sort described above as you can. Please save at least one document for EACH such DIFFERENT country.

If one document discusses several such countries, then you need not save other documents that repeat those, since your goal is to identify as many DIFFERENT countries of the sort described above as possible.

Instance#	Instance gloss
1	Poland
2	Singapore
3	Malaysia
4	Indonesia
5	Johor
6	Brunei
7	England (Cornwall)
8	Greece
9	United Kingdom (UK)
10	Egypt
11	US
12	Kenya
13	Cyprus(North and South)
14	Santo Domingo - Hispaniola - Dominican Republic
15	Scotland (Glasgow)
16	Jerusalem (Israel-Palestine)
17	Cuba
18	Monaco
19	Madeira
20	Greece
21	South Africa
22	Austria (Vienna)
23	New Zealand
24	Bahrain
25	Dubai
26	Wales
27	Mauritius
28	Britain
29	Northern Ireland
30	Sri Lanka
31	Ecuador
32	Israel
33	Australia
34	Jordan
35	Portugal
36	Slovenia
37	Czech Republic
38	Germany (Baden-Wurttemberg)
39	Spain
40	Tunisia
41	Shanghai
42	US (Florida)

- 43 Bermuda
- 44 Jamaica
- 45 Malta
- 46 India
- 47 Norway
- 48 Korea
- 49 Uganda
- 50 Ireland
- 51 Hong Kong
- 52 Vietnam
- 53 Sweden
- 54 Bulgaria
- 55 Slovakia
- 56 Zimbabwe

Number:
446i

Title:
tourists, violence

Description:
In what countries have tourists been subject to
acts of violence causing bodily harm or death?

Instances:
In the time allotted, please find as many DIFFERENT countries of
the sort described above as you can. Please save at least one
document for EACH such DIFFERENT country.
If one document discusses several such countries, then you need
not save other documents that repeat those, since your goal
is to identify as many DIFFERENT countries of the sort described
above as possible.

Instance#	Instance gloss
1	Angola
2	Ivory Coast
3	Kenya
4	Egypt
5	Mexico
6	Turkey
7	Great Britain (GB)
8	USA
9	Jamaica/GB
10	Thailand
11	China
12	Greece
13	Morocco
14	India
15	Cambodia
16	Israel

TREC-8 Interactive Track
Summary measures by search

[illegible]

Site	Searcher	Topic	System	#docs saved	Instance Prec. Rec.	Time (secs)	Instance coverage vector (or "0" if no docs saved) One bit/possible-instance; leftmost=#1; "1"=covered
BrkInt	P7_S2	428i	C	10	0.700 0.385	1080	0010100011111000000010001
BrkInt	P7_S1	431i	C	10	0.800 0.250	1200	101100000000000110111001000000010000000
BrkInt	P7_S5	408i	Z	7	1.000 0.333	840	01111011000000000100001
BrkInt	P7_S6	438i	Z	6	1.000 0.179	1200	011111000010000000000001000001000000100000000000000
BrkInt	P7_S4	446i	Z	8	0.625 0.312	900	0011011000010000
BrkInt	P8_S2	414i	C	6	0.667 0.833	1200	11111110110
BrkInt	P8_S1	428i	C	8	0.750 0.308	1200	00100000110000000001111
BrkInt	P8_S3	431i	C	9	0.889 0.475	1200	11111011100000001011000100100001111000
BrkInt	P8_S6	408i	Z	14	0.786 0.583	1200	111111100001110100001
BrkInt	P8_S4	438i	Z	13	0.769 0.232	1200	011111000
BrkInt	P8_S5	446i	Z	6	1.000 0.438	1200	10110110100000
BrkInt	P9_S5	414i	C	15	0.333 0.917	1200	1111111011011
BrkInt	P9_S6	428i	C	10	0.700 0.269	1200	1100000011000
BrkInt	P9_S4	431i	C	6	0.833 0.325	1200	1011000
BrkInt	P9_S3	408i	Z	12	1.000 0.583	1200	011111100001101110000
BrkInt	P9_S2	438i	Z	6	0.833 0.089	1200	000
BrkInt	P9_S1	446i	Z	7	0.714 0.312	1200	0011010100010000
NMSU	T1P1	408i	FULL	17	0.706 0.625	1200	01001110001111100111011
NMSU	T2P1	414i	FULL	5	0.600 0.750	1200	11111110001
NMSU	T6P1	446i	FULL	10	0.200 0.125	1200	000101000000000
NMSU	T3P1	428i	SUM	15	0.600 0.500	1200	1010000001111101000011110
NMSU	T4P1	431i	SUM	15	0.733 0.450	1200	111101110000001011100000010000000000000000000000000
NMSU	T5P1	438i	SUM	18	0.667 0.214	1200	000000000010010000000000000000000000000000000000000
NMSU	T1P10	408i	FULL	15	0.667 0.458	1200	010001100110000110001
NMSU	T4P10	431i	FULL	7	1.000 0.300	1200	1011010110100000100110000001000000000000000000000000
NMSU	T6P10	446i	FULL	10	0.300 0.188	1200	0011010000000000
NMSU	T3P10	414i	SUM	4	0.750 0.750	1200	11111110001
NMSU	T2P10	428i	SUM	12	0.593 0.385	1200	001000001111101000010000
NMSU	T5P10	438i	SUM	7	1.000 0.125	1200	000000000010010000000000000000000000000000000000000
NMSU	T2P11	414i	FULL	6	0.500 0.250	1200	11000000001
NMSU	T3P11	428i	FULL	10	0.600 0.346	1200	010000001111101000010000
NMSU	T5P11	438i	FULL	16	0.750 0.214	1200	000000000001001000000000000000000000000000000000000
NMSU	T1P11	408i	SUM	14	0.786 0.542	1200	0100111000011000011000110111
NMSU	T4P11	431i	SUM	5	1.000 0.275	1200	101100011010000010011000000000000000000000000000000
NMSU	T6P11	446i	SUM	6	0.500 0.188	1200	001101000000000
NMSU	T1P12	408i	FULL	18	0.722 0.625	1200	0111110110110001110001
NMSU	T2P12	414i	FULL	5	0.800 0.917	1200	11111110111
NMSU	T6P12	446i	FULL	7	0.429 0.250	1200	0011010100000000
NMSU	T3P12	428i	SUM	6	0.500 0.115	1200	0011000001000
NMSU	T4P12	431i	SUM	21	0.714 0.575	1200	00010001100001011011101100110111011011011011011011010
NMSU	T5P12	438i	SUM	40	0.725 0.589	1200	11111010101010000001011011110110110110110101010000001100
NMSU	T1P2	408i	FULL	16	0.750 0.542	1200	01000110011010000110111
NMSU	T3P2	428i	FULL	13	0.692 0.538	1200	101000010111101000011110
NMSU	T4P2	431i	FULL	10	1.000 0.475	1200	111110110100000101110000001000000000000000000000000
NMSU	T2P2	414i	SUM	4	1.000 0.917	1200	11111110111
NMSU	T5P2	438i	SUM	12	0.833 0.196	1200	011000000001001000000000000000000000000000000000000
NMSU	T6P2	446i	SUM	9	0.333 0.188	1200	00010100000000
NMSU	T2P3	414i	FULL	5	0.000 0.000	1200	000000000000
NMSU	T5P3	438i	FULL	31	0.548 0.304	1200	100000000100100100010100000101110100001110000000000000000
NMSU	T1P3	408i	SUM	18	0.339 0.312	1200	10010100000100010
NMSU	T3P3	428i	SUM	3	0.333 0.167	1200	01001011000
NMSU	T4P3	431i	SUM	18	0.444 0.308	1200	1011000001000001100010000
NMSU	T2P4	414i	SUM	22	0.682 0.575	1200	10110301111010000101111000001100001110010
NMSU	T3P4	428i	FULL	5	0.600 0.250	1200	110000000001
NMSU	T1P4	408i	FULL	9	0.778 0.500	1200	1010000101111101000001110
NMSU	T4P4	431i	FULL	22	0.864 0.357	1200	111000000000110000000000000000000000000000000000000
NMSU	T5P4	438i	FULL	16	0.750 0.583	1200	01001100011010000110111
NMSU	T1P4	408i	SUM	23	0.783 0.525	1200	101100011110000010111000000100000000000000000000000
NMSU	T4P4	431i	SUM	15	0.467 0.375	1200	10110100000100010

A-223

[illegible]

Site	Search	Searcher	Topic	System	#docs		Instance Prec.	Time (secs)	Instance coverage vector (or "0" if no docs saved)
					saved	Rec.			
RU-INT	s036a_414i	s036a	414i	RF	3	1.000	0.250	1136	1100000000001
	s036a_431i	s036a	431i	RF	5	1.000	0.375	1241	11111100000000000101110000001000001111000
	s036a_438i	s036a	438i	RF	13	1.000	0.232	1186	0000000101001000100000000010110001010000010011000000001000
	s01_408	s01	408i	Okapi_norF	13	0.846	0.542	1035	0111111001010001100001100001
	s01_438	s01	438i	Okapi_norF	19	0.895	0.504	1237	00000000110100001000001000100010100001000011110000111000
	s01_446	s01	446i	Okapi_norF	8	0.375	0.188	1241	0011010000000000
	s01_414	s01	414i	Okapi_withRF	3	1.000	0.333	1059	010000000111
	s01_428	s01	428i	Okapi_withRF	11	0.818	0.462	1275	00111000001111010000110001
	s01_431	s01	431i	Okapi_withRF	9	0.778	0.425	1275	1011000110000001110011000000100011111000
	s02_414	s02	414i	Okapi_norF	5	0.800	0.917	1320	111111111110
sheff	s02_428	s02	428i	Okapi_norF	11	0.727	0.500	1195	00101000010111001000001000111110
	s02_446	s02	446i	Okapi_norF	9	0.333	0.250	1217	0010101000000000
	s02_408	s02	408i	Okapi_withRF	12	0.750	0.333	1271	0100011000001100001110000
	s02_431	s02	431i	Okapi_withRF	9	1.000	0.400	1191	11111101100000010111000000100110000000
	s02_438	s02	438i	Okapi_withRF	12	1.000	0.214	1159	000000001101000010000010001000000100110000000
	s03_408	s03	408i	Okapi_norF	10	0.700	0.375	1129	01001111000011000011000010000
	s03_438	s03	438i	Okapi_norF	14	0.786	0.196	1042	000000100000000000001010000101001000101000000010000001000
	s03_446	s03	446i	Okapi_norF	11	0.273	0.188	1074	0011010000000000
	s03_414	s03	414i	Okapi_withRF	6	0.500	0.833	1355	111111111001
	s03_428	s03	428i	Okapi_withRF	11	0.727	0.423	1244	0010000111111001000110001
sheff	s03_431	s03	431i	Okapi_withRF	7	1.000	0.350	1263	111111011010000010111000000000000000
	s04_408	s04	408i	Okapi_norF	10	0.300	0.250	1206	010010110000001100000000
	s04_438	s04	438i	Okapi_norF	13	0.538	0.143	1355	111000000000000000000010000001000000000000000000000
	s04_446	s04	446i	Okapi_norF	3	0.667	0.188	1207	0011010000000000
	s04_414	s04	414i	Okapi_withRF	7	0.429	0.667	1231	111111110000
	s04_428	s04	428i	Okapi_withRF	8	0.875	0.385	1218	00101000010111001001010000
	s04_431	s04	431i	Okapi_withRF	12	0.833	0.450	1131	1010000110100000110111000000100011111000
	s05_408	s05	408i	Okapi_norF	22	0.818	0.708	1237	1111111100101110110110001
	s05_414	s05	414i	Okapi_norF	4	1.000	0.917	666	111111110111
	s05_431	s05	431i	Okapi_norF	19	0.842	0.550	1453	1111110111100000110011001001000001111000
sheff	s05_428	s05	428i	Okapi_withRF	10	0.900	0.423	1297	0010100011111001000110000
	s05_438	s05	438i	Okapi_withRF	13	1.000	0.232	1358	000000001101001000000100010010010100001100000010000001000
	s05_446	s05	446i	Okapi_withRF	10	0.700	0.438	1115	1011010100100010
	s06_408	s06	408i	Okapi_norF	5	0.800	0.292	1440	01110110000000001000000
	s06_438	s06	438i	Okapi_norF	7	0.714	0.089	1150	000
	s06_446	s06	446i	Okapi_norF	6	0.167	0.125	1154	0011000000000000
	s06_414	s06	414i	Okapi_withRF	3	0.667	0.667	1098	111111110000
	s06_428	s06	428i	Okapi_withRF	8	0.875	0.385	1305	00100000101111001000110001
	s06_431	s06	431i	Okapi_withRF	1	1.000	0.150	1215	101100000000000001001100000000000000000000
	s07_414	s07	414i	Okapi_norF	4	0.750	0.750	1315	111111110001
sheff	s07_428	s07	428i	Okapi_norF	10	0.900	0.346	1232	01111000001000001100010001
	s07_431	s07	431i	Okapi_norF	8	1.000	0.476	1270	1111110110100000101110000001000001111000
	s07_408	s07	408i	Okapi_withRF	7	1.000	0.292	1292	010011010000010000000001
	s07_438	s07	438i	Okapi_withRF	11	0.909	0.179	1090	00000000000000000000010101001100000100000001010000001000000101000
	s07_446	s07	446i	Okapi_withRF	5	0.800	0.250	1268	0010000100100010
	s08_408	s08	408i	Okapi_norF	7	1.000	0.375	997	010010110000000000101011
	s08_438	s08	438i	Okapi_norF	13	0.923	0.214	1052	0000000000100
	s08_446	s08	446i	Okapi_norF	5	0.600	0.188	1050	00010100000000
	s08_414	s08	414i	Okapi_withRF	8	0.500	0.917	1204	111111110111
	s08_428	s08	428i	Okapi_withRF	8	0.750	0.231	1127	0010000000100000010000010001
sheff	s08_431	s08	431i	Okapi_withRF	11	0.909	0.450	1104	1010000110001100001100110000001000011111000
	s09_414	s09	414i	Okapi_norF	10	0.400	1.000	1059	111111111111
	s09_428	s09	428i	Okapi_norF	5	0.800	0.154	1137	00101000001000000000000000
	s09_431	s09	431i	Okapi_norF	14	0.714	0.400	1166	001000011000000010011000000010000001111000
	s09_408	s09	408i	Okapi_withRF	7	0.714	0.167	1119	000000100000001000000000000000000000000000000000000
	s09_438	s09	438i	Okapi_withRF	11	1.000	0.268	1188	0111100010000100000001000110100100000000000000000000000000000000
	s09_446	s09	446i	Okapi_withRF	3	1.000	0.125	921	0001010000000000
	s10_414	s10	414i	Okapi_norF	6	0.833	0.917	1077	111111110111
	s10_428	s10	428i	Okapi_norF	11	0.727	0.423	1137	001010000111110010000010001

A-231

Site	Search	Searcher	Topic	System	#docs saved	Instance Prec. Rec.	Time (secs)	Instance coverage vector (or "0" if no docs saved) One bit/possible-instance; leftmost=#1; "1"=covered
unc8iap	23_446	iris23p	4461	iriss	10	0.600 0.375	1200	00110110000011000
unc8iap	24_408	iris24p	4081	irisp	5	0.800 0.208	1200	0100001000110000000000001
unc8iap	24_431	iris24p	4311	irisp	4	1.000 0.100	1200	000000000001000000000010111000000000000000
unc8iap	24_446	iris24p	4461	irisp	4	0.000 0.000	1200	00
unc8iap	24_414	iris24p	4141	iriss	4	0.750 0.500	1200	01100000111
unc8iap	24_428	iris24p	4281	iriss	8	0.625 0.192	1200	0011000101000000100000000000000000000000
unc8iap	24_438	iris24p	4381	iriss	4	0.750 0.054	1200	0000000000000000100000000000000000000000000000000
unc8iap	2_408	iris2p	4081	irisp	4	0.500 0.083	1200	010001000000000000000000000000000000000000
unc8iap	2_428	iris2p	4281	irisp	7	0.571 0.154	1200	10100000010000000000000000000000000000000001
unc8iap	2_438	iris2p	4381	irisp	4	0.750 0.054	1200	000
unc8iap	2_414	iris2p	4141	iriss	7	0.429 0.917	1200	11111110111
unc8iap	2_431	iris2p	4311	iriss	6	0.833 0.125	1200	00010000001000000000110010000000000000000000
unc8iap	2_446	iris2p	4461	iriss	7	0.429 0.125	1200	0001010000000000
unc8iap	3_408	iris3p	4081	irisp	25	0.800 0.500	1200	0100111001011100100001
unc8iap	3_414	iris3p	4141	irisp	5	0.800 0.750	1200	11111110001
unc8iap	3_428	iris3p	4281	irisp	10	0.900 0.500	1200	00101000100111101000111110
unc8iap	3_431	iris3p	4311	iriss	9	0.889 0.400	1200	11111101000100000100110000001000001111000
unc8iap	3_438	iris3p	4381	iriss	14	0.643 0.143	1200	00000000000000010000000000110000000010000001000000100010
unc8iap	3_446	iris3p	4461	iriss	29	0.448 0.125	1200	0011000000000000
unc8iap	4_428	iris4p	4281	irisp	6	1.000 0.346	1200	001100001001110010001000001
unc8iap	4_431	iris4p	4311	irisp	2	1.000 0.125	1200	0001000000000000011110000000000000000000000000000
unc8iap	4_438	iris4p	4381	irisp	4	0.500 0.036	1200	000
unc8iap	4_408	iris4p	4081	iriss	5	1.000 0.208	1200	01001011000
unc8iap	4_414	iris4p	4141	iriss	4	0.750 0.250	1200	110000000001
unc8iap	4_446	iris4p	4461	iriss	3	0.667 0.125	1200	000101000
unc8iap	5_428	iris5p	4281	irisp	4	0.250 0.038	1200	00100
unc8iap	5_431	iris5p	4311	irisp	9	0.889 0.200	1200	0001000000010000101110010010000000000000000000000
unc8iap	5_446	iris5p	4461	irisp	8	0.250 0.125	1200	00000100000000010
unc8iap	5_408	iris5p	4081	iriss	0	0.000 0.000	1200	0
unc8iap	5_414	iris5p	4141	iriss	0	0.000 0.000	1200	0
unc8iap	5_438	iris5p	4381	iriss	0	0.000 0.000	1200	0
unc8iap	6_408	iris6p	4081	irisp	13	0.692 0.292	1200	1100101100000100000000000000000000000000000000000
unc8iap	6_431	iris6p	4311	irisp	5	0.800 0.225	1200	1011000100000000010011100000000001000000000000000000000
unc8iap	6_446	iris6p	4461	irisp	5	0.600 0.125	1200	0101000000000000
unc8iap	6_414	iris6p	4141	iriss	3	1.000 0.750	1200	11111110001
unc8iap	6_428	iris6p	4281	iriss	6	0.667 0.269	1200	001000010111100000000000
unc8iap	6_438	iris6p	4381	iriss	3	0.333 0.018	1200	0000000000000100000000000000000000000000000000000
unc8iap	7_428	iris7p	4281	irisp	6	0.500 0.192	1200	00100
unc8iap	7_438	iris7p	4381	irisp	10	0.700 0.125	1200	00000001000001001000010000011000100000000000000000000000000000
unc8iap	7_446	iris7p	4461	irisp	20	0.450 0.188	1200	0011000100000000
unc8iap	7_408	iris7p	4081	iriss	18	0.722 0.292	1200	0100101100110000001000000
unc8iap	7_414	iris7p	4141	iriss	11	0.545 0.750	1200	11111110001
unc8iap	7_431	iris7p	4311	iriss	8	0.500 0.100	1200	0001000000001000000001000000010000000000000000000
unc8iap	8_414	iris8p	4141	irisp	3	0.667 0.167	1200	0100000001
unc8iap	8_431	iris8p	4311	irisp	16	0.750 0.500	1200	1011000101001000110011001001101101111000
unc8iap	8_446	iris8p	4461	irisp	10	0.300 0.125	1200	0001000100000000
unc8iap	8_408	iris8p	4081	iriss	14	0.786 0.458	1200	010011110000110001110001
unc8iap	8_428	iris8p	4281	iriss	5	0.800 0.154	1200	00110000000000010000000001
unc8iap	8_438	iris8p	4381	iriss	7	0.714 0.089	1200	0000000000000000000001010100000000000000000000000000000
unc8iap	9_431	iris9p	4311	irisp	3	0.667 0.050	1200	00000001000
unc8iap	9_438	iris9p	4381	irisp	3	0.333 0.018	1200	000
unc8iap	9_446	iris9p	4461	irisp	8	0.750 0.375	1200	0011011000011000
unc8iap	9_408	iris9p	4081	iriss	10	0.900 0.333	1200	010001100010110001100000
unc8iap	9_414	iris9p	4141	iriss	3	0.667 0.667	1200	11111110000
unc8iap	9_428	iris9p	4281	iriss	7	0.714 0.308	1200	0011000011111100000000000000

Query Track

5 groups participated in the Query Track for TREC 8. Each group submitted one or more querysets consisting of 50 queries. There were a total of 23 querysets (Table 2) in 4 different categories.

- Short: 2-4 words
- Sentence: 1 sentence, normally less than 1 line.
- Sentence-Rel: 1 sentence, based on relevant documents only.
- Weighted Terms: A long (about 100 terms) list of weighted terms.

Each group then ran one or more retrieval approaches on each of the natural language querysets (the first 3 above) and on the weighted terms if they could do so easily. There were 9 runsets. Two groups did not run the two weighted term categories, thus there were a total of 203 runs evaluated ($9 * 23 - 4$). (Table 1).

RunSet	Group	Approach
APL	Johns Hopkins	APL system
INQa	UMass	INQUERY, words only
INQe	UMass	INQUERY, words + structure + expansion
INQp	UMass	INQUERY, words + structure
Saba	Sabir Research	SMART, words only
Sabe	Sabir Research	SMART, words + full expansion
Sabm	Sabir Research	SMART, words + modest expansion
acs	ACSys	PADRE
pir	Queens College, CUNY	PIRCS

Table 1: 9 Runsets

QuerySet	Group	Category
INQ1a	UMass	Short
INQ1b	UMass	Short
INQ1c	UMass	Short
INQ1d	UMass	Short
INQ1e	UMass	Short
INQ2a	UMass	Sentence
INQ2b	UMass	Sentence
INQ2c	UMass	Sentence
INQ2d	UMass	Sentence
INQ2e	UMass	Sentence
INQ3a	UMass	Sentence-Rel
INQ3b	UMass	Sentence-Rel
INQ3c	UMass	Sentence-Rel
INQ3d	UMass	Sentence-Rel
INQ3e	UMass	Sentence-Rel
Sab1a	Sabir	Short
Sab1b	Sabir	Short
Sab1c	Sabir	Short
Sab3a	Sabir	Sentence-Rel
acs1a	ACSys	Short
pir1a	Queens	Short
APL5a	Johns Hopkins	Weighted Terms
APL5b	Johns Hopkins	Weighted Terms

Table 2: 23 Querysets

Query Track

For each query, average the scores of the 9 runs. Then for each queryset, average these 50 average query scores to get a notion of queryset variability (whatever that means). Also report standard deviation of these average query scores.

Note that runs APL5a and APL5b have low averages since two runsets didn't do them and have 0.0 averaged in.

QuerySet	Average	Standard Deviation
INQ1a	0.1810	0.2091
INQ1b	0.2134	0.2111
INQ1c	0.2292	0.2190
INQ1d	0.1923	0.2193
INQ1e	0.2278	0.2523
INQ2a	0.1655	0.1813
INQ2b	0.1907	0.1722
INQ2c	0.2434	0.2132
INQ2d	0.2054	0.2005
INQ2e	0.2402	0.2079
INQ3a	0.1348	0.1581
INQ3b	0.1215	0.1530
INQ3c	0.1325	0.1544
INQ3d	0.1602	0.1827
INQ3e	0.1791	0.1943
Sab1a	0.2405	0.2311
Sab1b	0.2520	0.2214
Sab1c	0.2549	0.2284
Sab3a	0.2389	0.1968
acs1a	0.2455	0.2271
pir1a	0.2488	0.2058
APL5a	0.2073	0.1608
APL5b	0.2254	0.1230

Table 3: Queryset Averages

Query Track

Take the average query scores over 9 runs for each query. Then for each topic, average the 23 queryset average query scores to get a notion of topic variability.

Query	Average	Standard Deviation	—	Query	Average	Standard Deviation
51	0.4489	0.1490		76	0.1152	0.0802
52	0.5058	0.2099		77	0.2795	0.1789
53	0.2410	0.1143		78	0.4231	0.3298
54	0.3580	0.1481		79	0.1331	0.0869
55	0.4743	0.1652		80	0.0602	0.0611
56	0.4097	0.2525		81	0.1105	0.0659
57	0.2894	0.1858		82	0.4436	0.1190
58	0.5716	0.1938		83	0.0830	0.0904
59	0.0992	0.1019		84	0.0237	0.0198
60	0.0660	0.0492		85	0.1361	0.0824
61	0.3270	0.1452		86	0.3546	0.1734
62	0.2729	0.0448		87	0.0587	0.0767
63	0.1411	0.0511		88	0.1779	0.1191
64	0.2183	0.0442		89	0.0767	0.0785
65	0.1544	0.0749		90	0.3215	0.1308
66	0.1990	0.1539		91	0.0331	0.0391
67	0.0827	0.0933		92	0.0362	0.0324
68	0.0931	0.1084		93	0.4242	0.0996
69	0.2006	0.1374		94	0.1201	0.0811
70	0.6614	0.0584		95	0.0343	0.0282
71	0.0664	0.0692		96	0.0561	0.0435
72	0.0669	0.0826		97	0.0642	0.0249
73	0.0400	0.0626		98	0.1791	0.0254
74	0.0134	0.0231		99	0.3045	0.0842
75	0.0174	0.0141		100	0.2156	0.1904

Table 4: Query Averages

Average the queries and querysets to get a notion of runset variability.

Runset	Average
APL	0.2162
INQa	0.1669
INQe	0.2295
INQp	0.1940
Saba	0.2051
Sabe	0.2440
Sabm	0.2240
acs	0.1470
pir	0.2244

Table 5: Runset Averages

Explanation of QA Results

The score for an individual question is the reciprocal of the rank at which the first correct response was found, or 0 if no correct response was found in the top 5 responses. The score for the run as a whole is the mean of the reciprocal ranks over the 198 test questions. Also reported is the number of questions for which no correct response was found.

The graph plots the score for a given run against the median score for each question. The median scores are computed separately over runs with a maximum answer length of 50 bytes and runs with a maximum answer length of 250 bytes. The median score is printed as a cross hatch, and the question ids along the x-axis are sorted by decreasing median score. An arrow runs from the median score to the score for the current run. If the run's score is greater than the median, the arrowhead is filled; otherwise the arrowhead is empty. If no arrow appears, then the run's score is identical to the median score for that question.

The order of the questions in the graphs is given below.

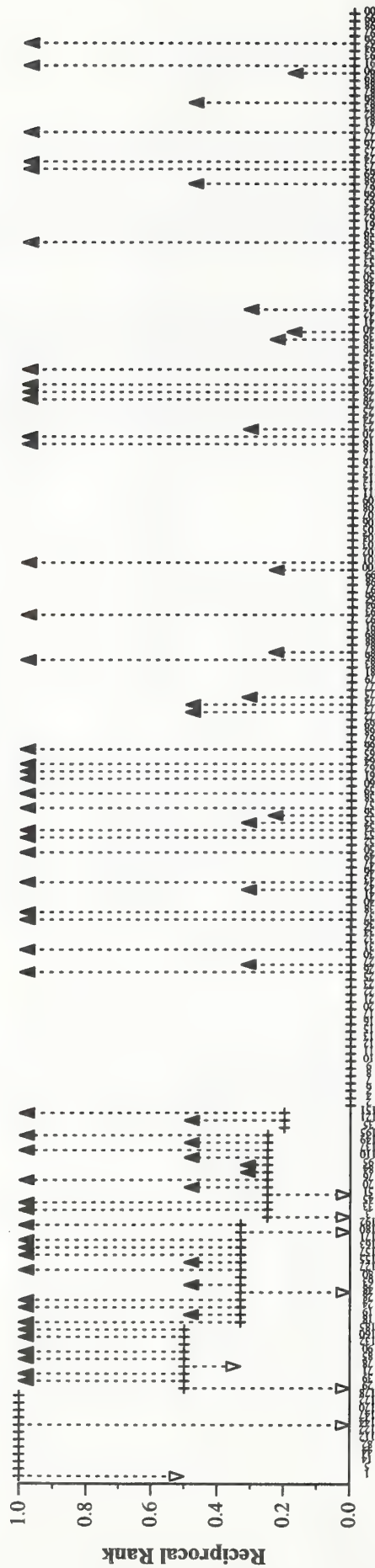
Table 1: Question number by decending median for runs with maximum answer length 50 bytes

1	5	14	44	82	112	122	144	147	170	172	178	29	39	71
78	85	90	132	160	185	18	19	24	28	48	63	80	127	155
157	163	171	180	192	3	33	45	51	70	76	84	95	110	137
149	195	35	121	151	2	4	6	7	8	9	10	11	12	13
15	16	17	20	21	22	23	25	26	27	30	31	32	34	36
37	38	40	41	42	43	46	47	49	50	52	53	54	55	56
57	58	59	60	61	62	64	65	66	67	68	69	72	73	74
75	77	79	81	83	86	87	88	89	91	92	93	94	96	97
98	99	100	101	102	103	104	105	106	107	108	109	111	113	114
115	116	117	118	119	120	123	124	125	126	128	129	130	133	134
135	136	138	139	140	141	142	143	145	146	148	150	152	153	154
156	158	159	161	162	164	165	166	167	168	169	173	174	175	176
177	179	181	182	183	186	187	188	189	190	191	193	194	196	197
198	199	200												

Table 2: Question number by decending median for runs with maximum answer length 250 bytes

1	13	14	16	18	28	29	30	39	44	45	46	48	50	58
63	66	80	82	90	110	112	117	118	122	127	128	139	147	155
163	178	180	5	10	26	35	38	60	65	71	76	78	83	85
99	103	120	124	132	134	140	144	154	160	162	170	172	3	19
24	37	40	41	42	51	57	68	70	79	84	95	100	106	137
151	157	185	190	6	52	88	102	121	145	149	171	183	192	199
33	59	62	130	138	168	188	200	2	4	7	8	9	11	12
15	17	20	21	22	23	25	27	31	32	34	36	43	47	49
53	54	55	56	61	64	67	69	72	73	74	75	77	81	86
87	89	91	92	93	94	96	97	98	101	104	105	107	108	109
111	113	114	115	116	119	123	125	126	129	133	135	136	141	142
143	146	148	150	152	153	156	158	159	161	164	165	166	167	169
173	174	175	176	177	179	181	182	186	187	189	191	193	194	195
196	197	198												

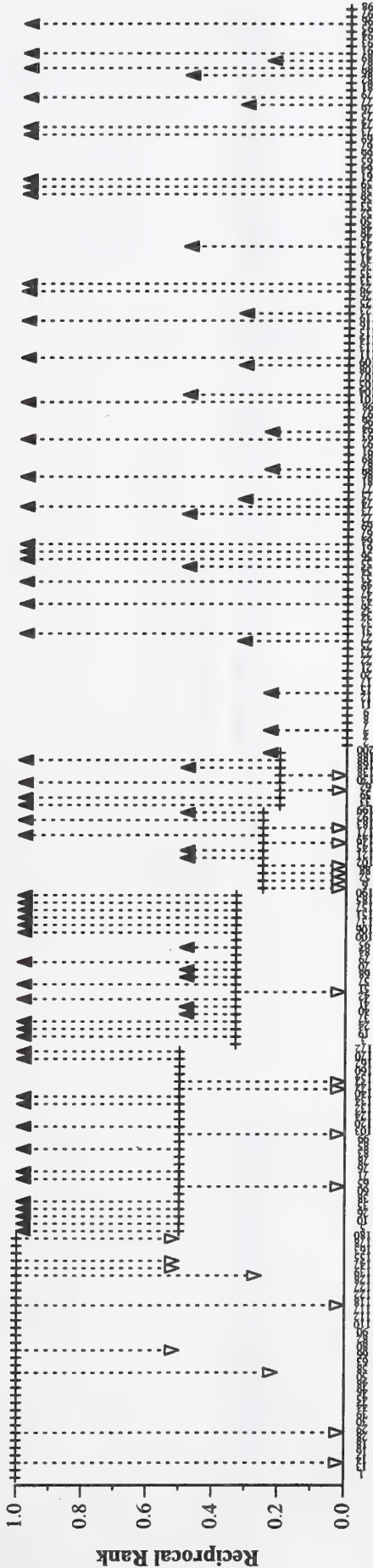
Run:	attqa50e
Answer length:	50
Mean reciprocal rank:	0.356
# questions with no answer found:	109



Question number by descending median

Run scores vs. median scores by question for run attqa50e

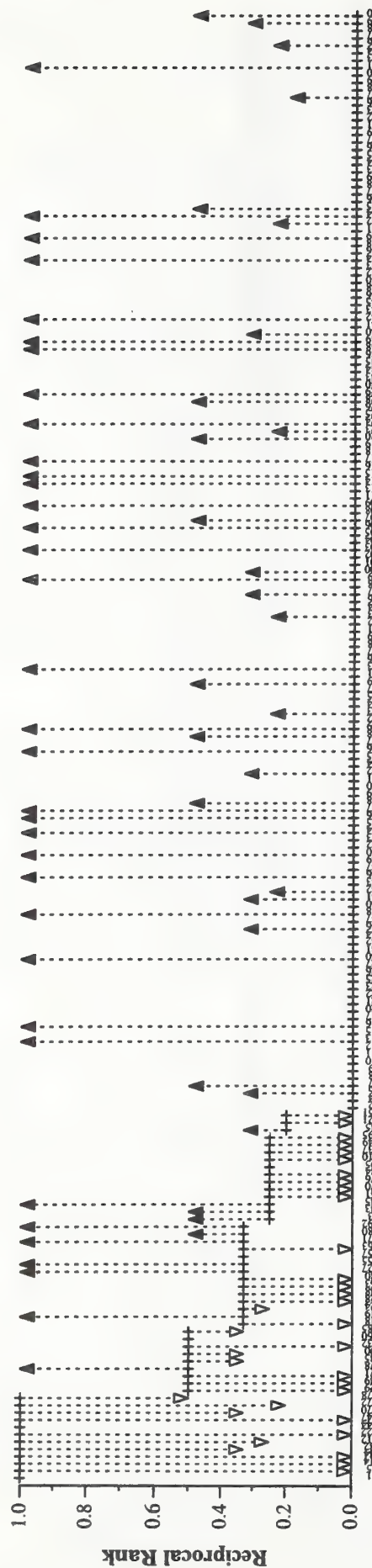
Run:	attqa250e
Answer length:	250
Mean reciprocal rank:	0.483
# questions with no answer found:	78



Question number by descending median

Run scores vs. median scores by question for run attqa250e

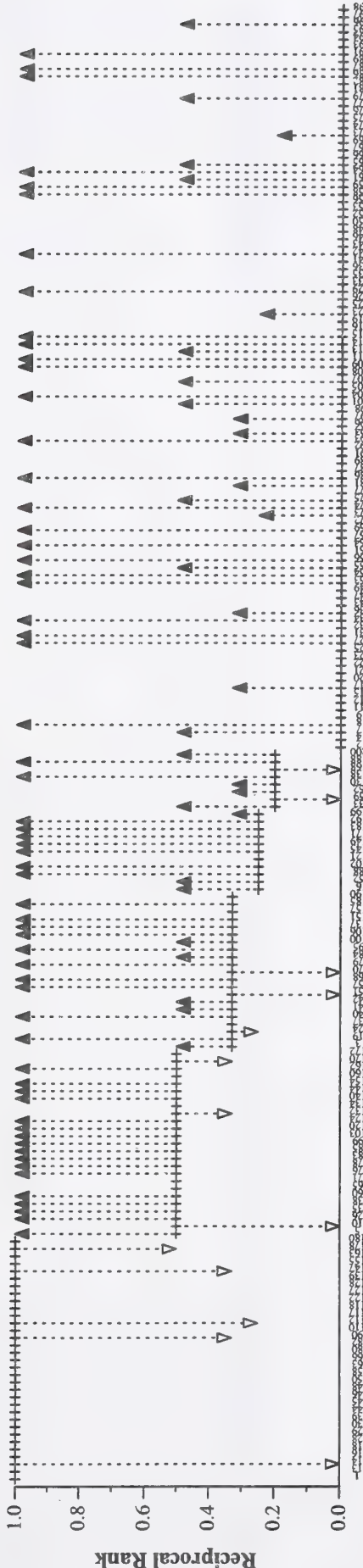
Run:	attqa50p
Answer length:	50
Mean reciprocal rank:	0.261
# questions with no answer found:	121



Question number by descending median

Run scores vs. median scores by question for run attqa50p

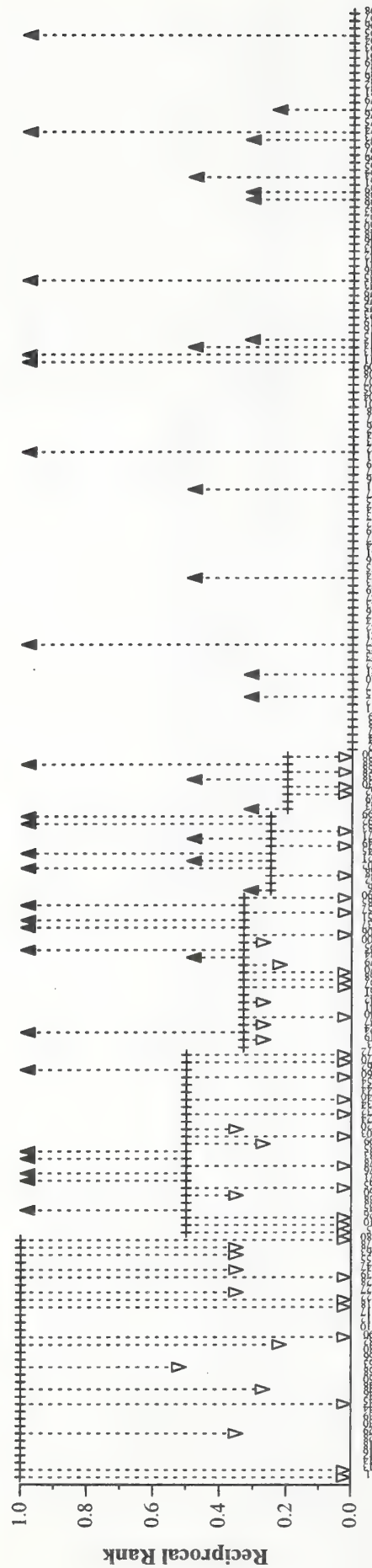
Run:	attqa250p
Answer length:	250
Mean reciprocal rank:	0.545
# questions with no answer found:	63



Question number by descending median

Run scores vs. median scores by question for run attqa250p

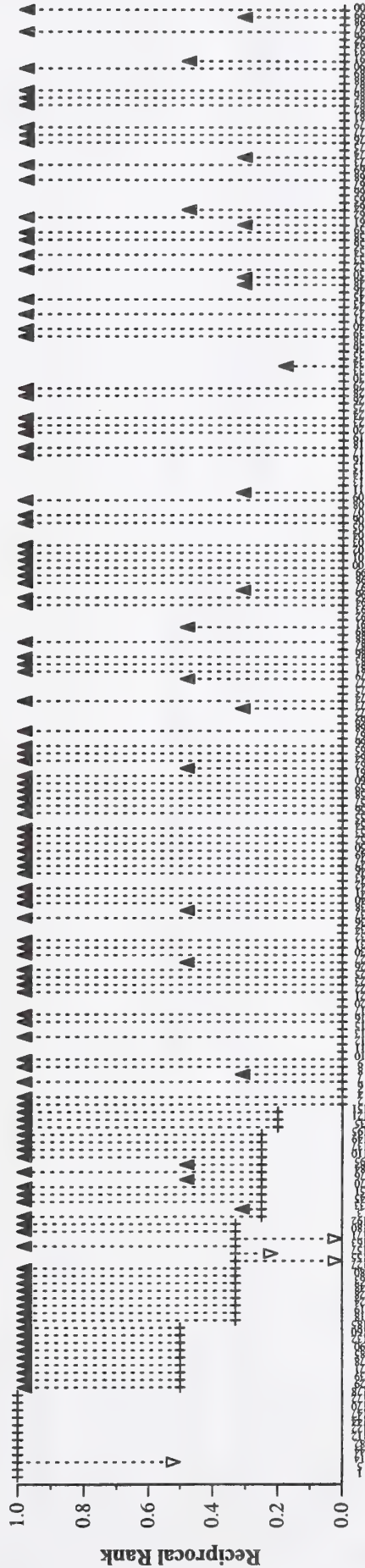
Run:	clr99s
Answer length:	250
Mean reciprocal rank:	0.281
# questions with no answer found:	115



Question number by descending median

Run scores vs. median scores by question for run clr99s

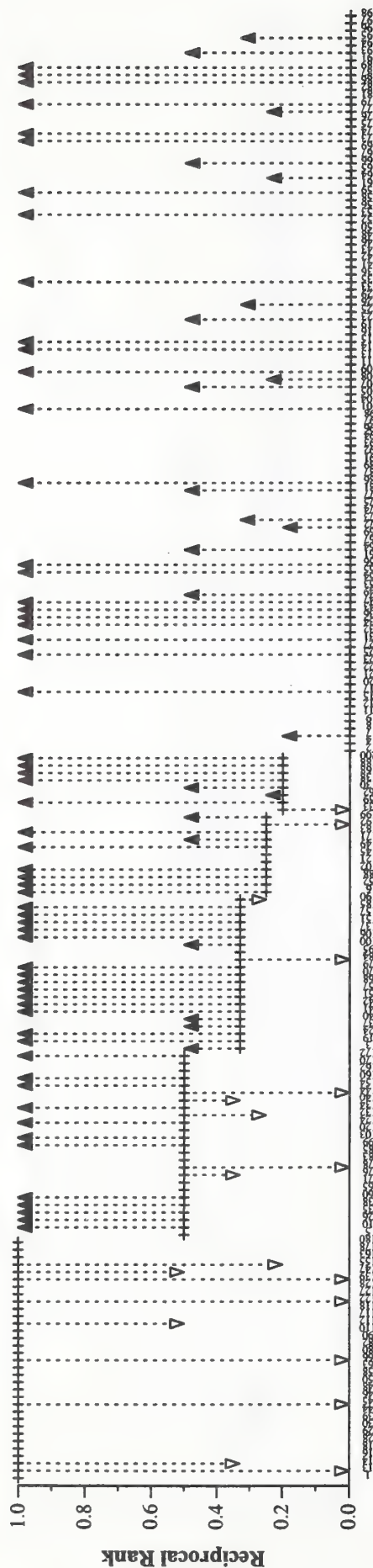
Run:	textextract9908
Answer length:	50
Mean reciprocal rank:	0.660
# questions with no answer found:	54



Question number by descending median

Run scores vs. median scores by question for run textextract9908

Run:	GePenn
Answer length:	250
Mean reciprocal rank:	0.510
# questions with no answer found:	72



Question number by descending median

Run scores vs. median scores by question for run GePenn

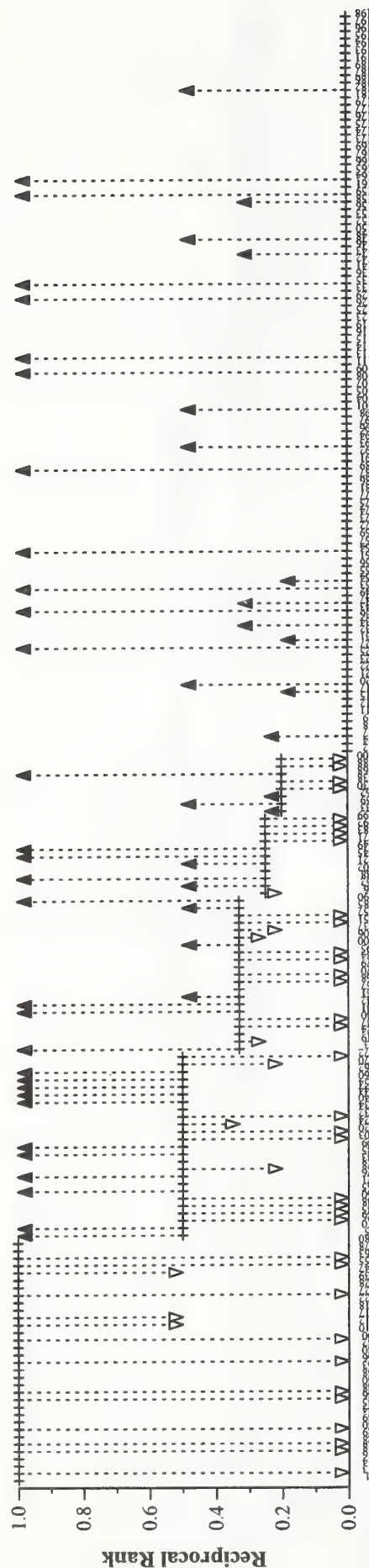
Run:	CRDBASE050
Answer length:	50
Mean reciprocal rank:	0.158
# questions with no answer found:	148



Question number by descending median

Run scores vs. median scores by question for run CRDBASE050

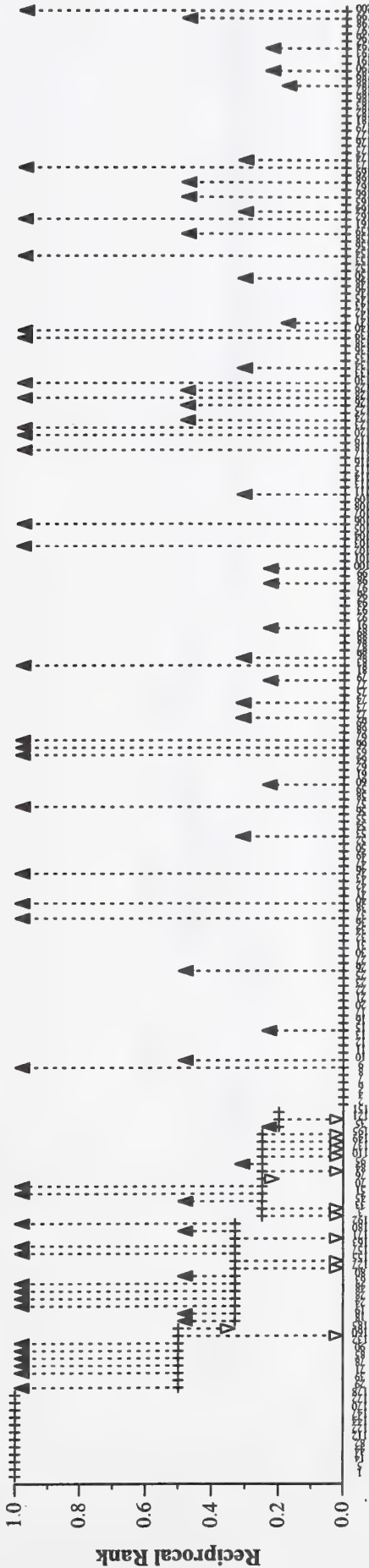
Run:	CRDBASE250
Answer length:	250
Mean reciprocal rank:	0.319
# questions with no answer found:	111



Question number by descending median

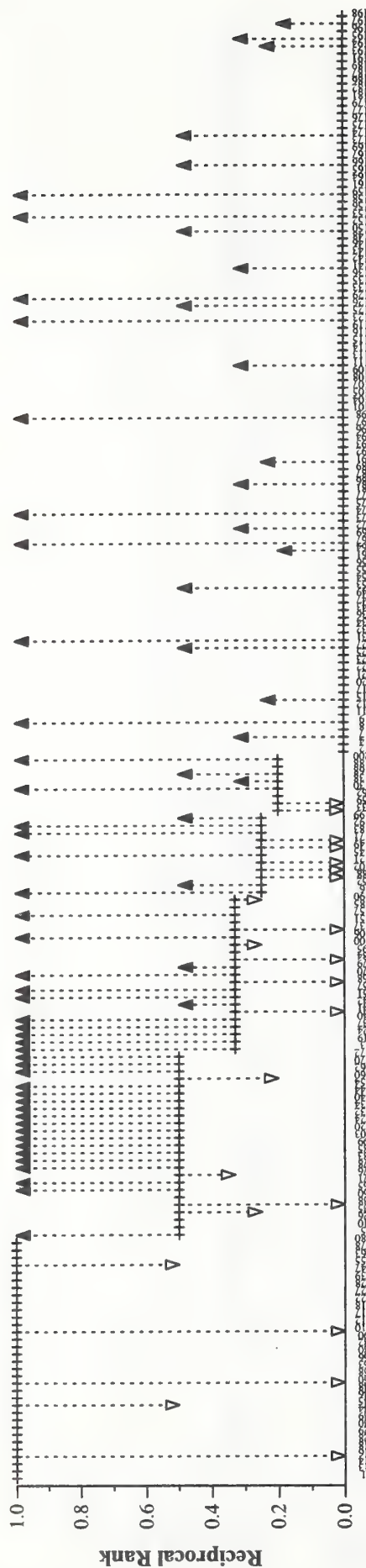
Run scores vs. median scores by question for run CRDBASE250

Run:	IBMDR995
Answer length:	50
Mean reciprocal rank:	0.319
# questions with no answer found:	110



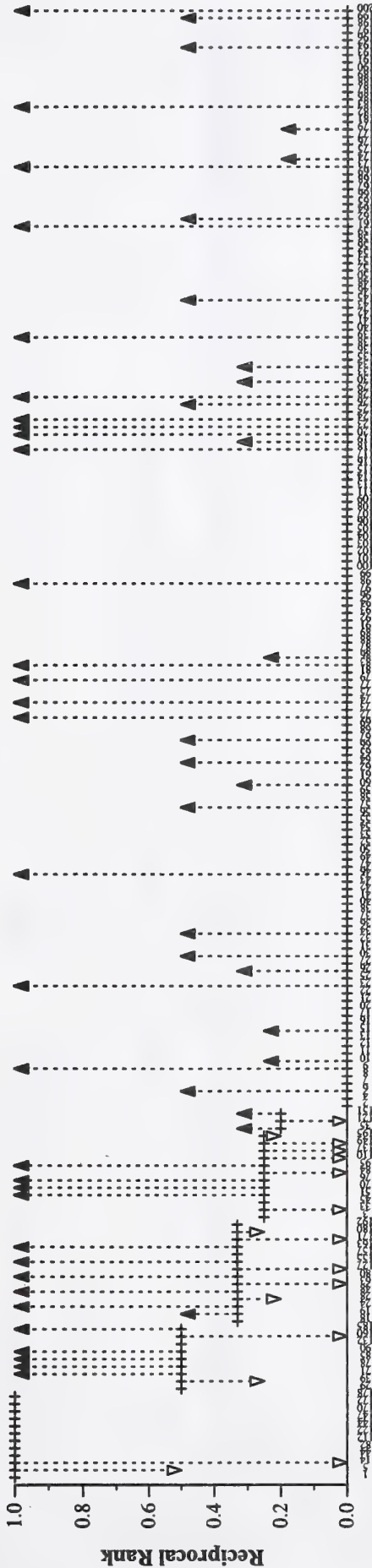
Question number by descending median
Run scores vs. median scores by question for run IBMDR995

Run:	IBMDR992
Answer length:	250
Mean reciprocal rank:	0.430
# questions with no answer found:	89

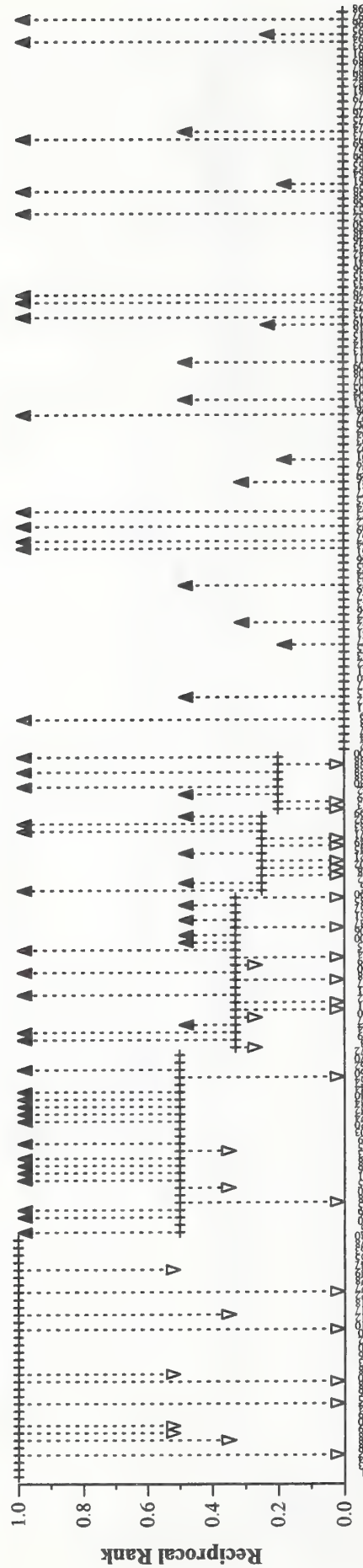


Question number by descending median
Run scores vs. median scores by question for run IBMDR992

Run:	IBMVS995
Answer length:	50
Mean reciprocal rank:	0.280
# questions with no answer found:	120



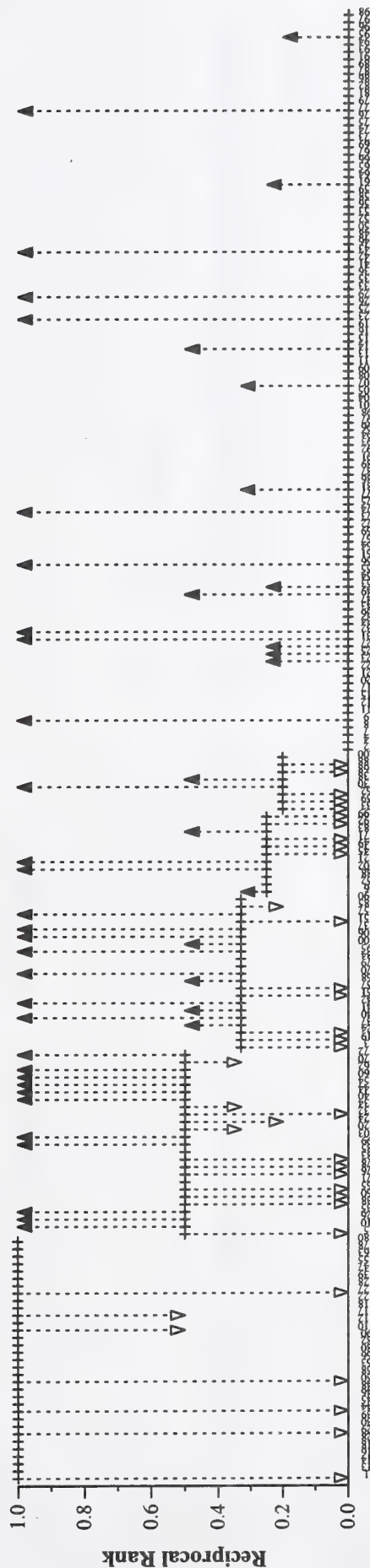
Run:	IBMVS992
Answer length:	250
Mean reciprocal rank:	0.395
# questions with no answer found:	95



Question number by descending median

Run scores vs. median scores by question for run IBMVS992

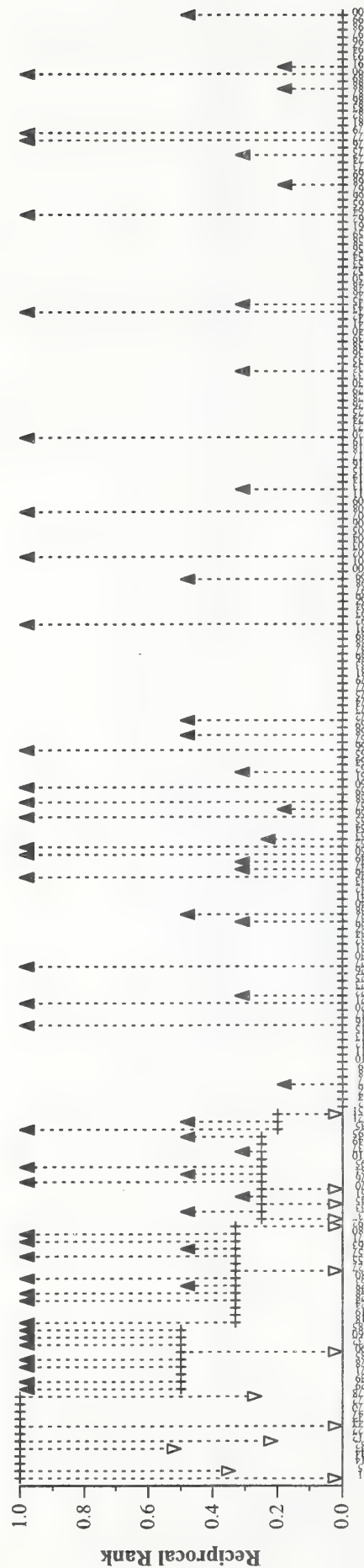
Run:	LimsiLC
Answer length:	250
Mean reciprocal rank:	0.341
# questions with no answer found:	110



Question number by descending median

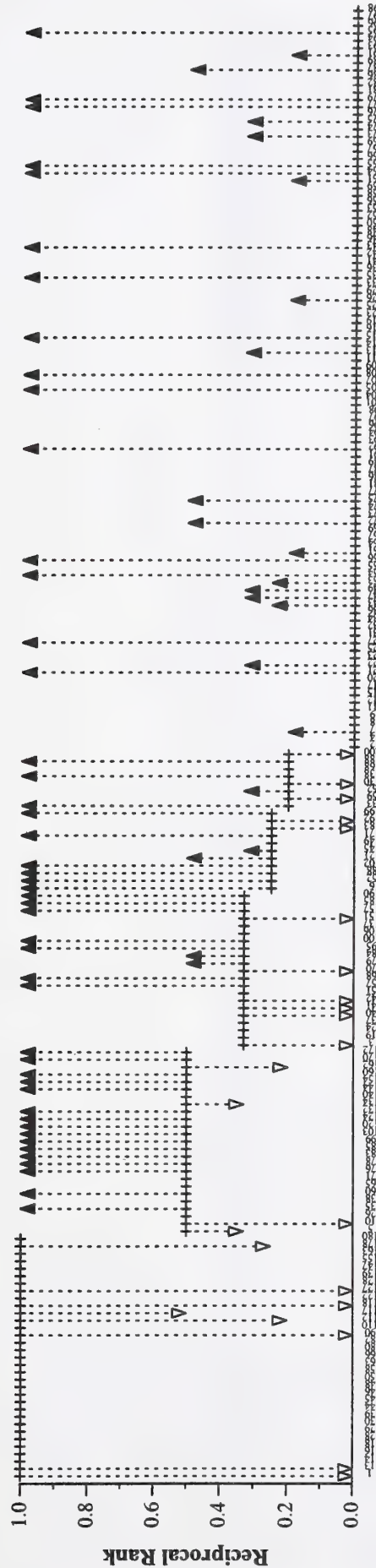
Run scores vs. median scores by question for run LimsiLC

Run:	MTR99050
Answer length:	50
Mean reciprocal rank:	0.281
# questions with no answer found:	118



Run scores vs. median scores by question for run MTR99050

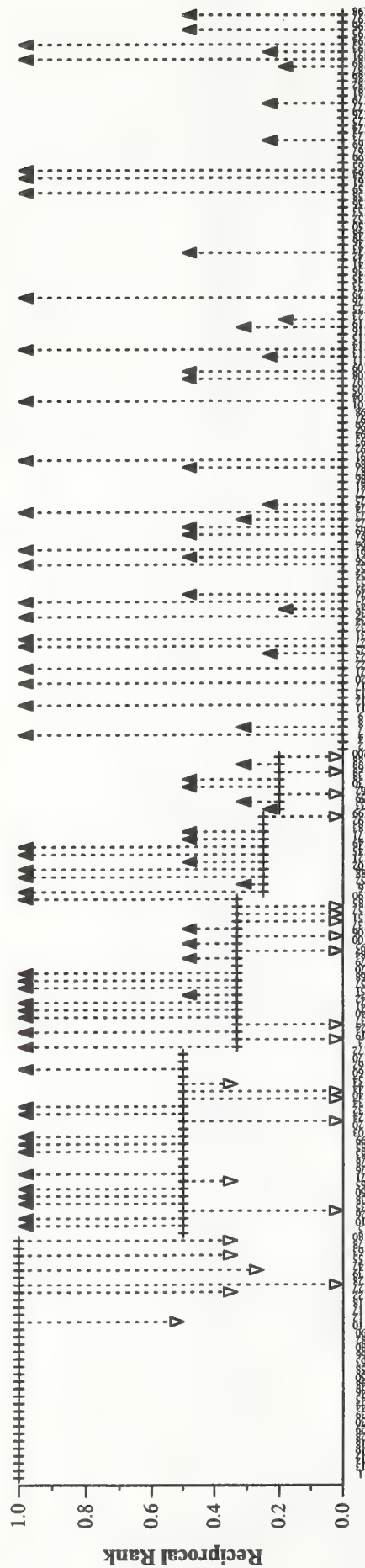
Run:	MTR99250
Answer length:	250
Mean reciprocal rank:	0.434
# questions with no answer found:	86



Question number by descending median

Run scores vs. median scores by question for run MTR99250

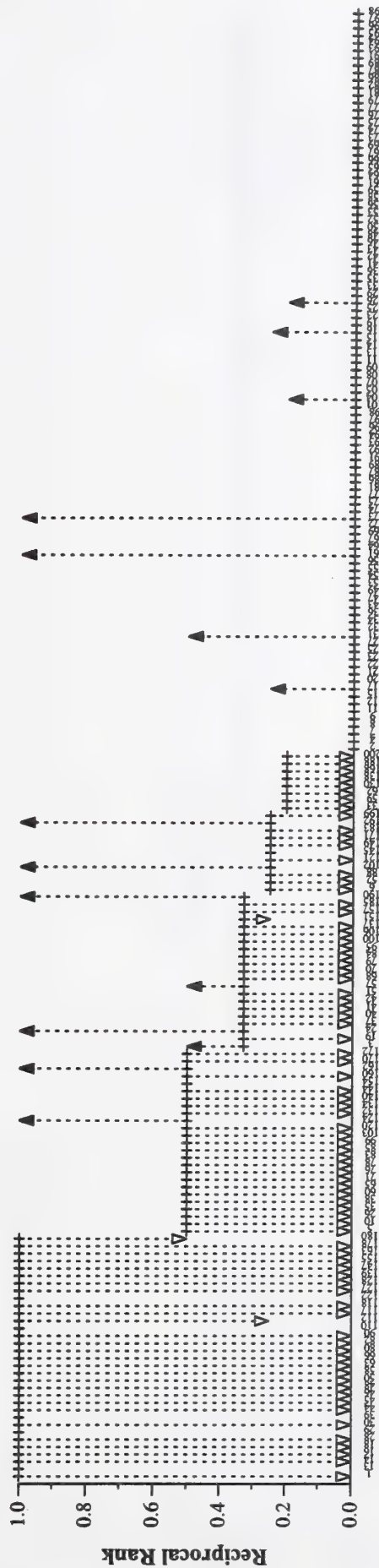
Run:	uwmt8qa1
Answer length:	250
Mean reciprocal rank:	0.471
# questions with no answer found:	74



Question number by descending median

Run scores vs. median scores by question for run uwmt8qa1

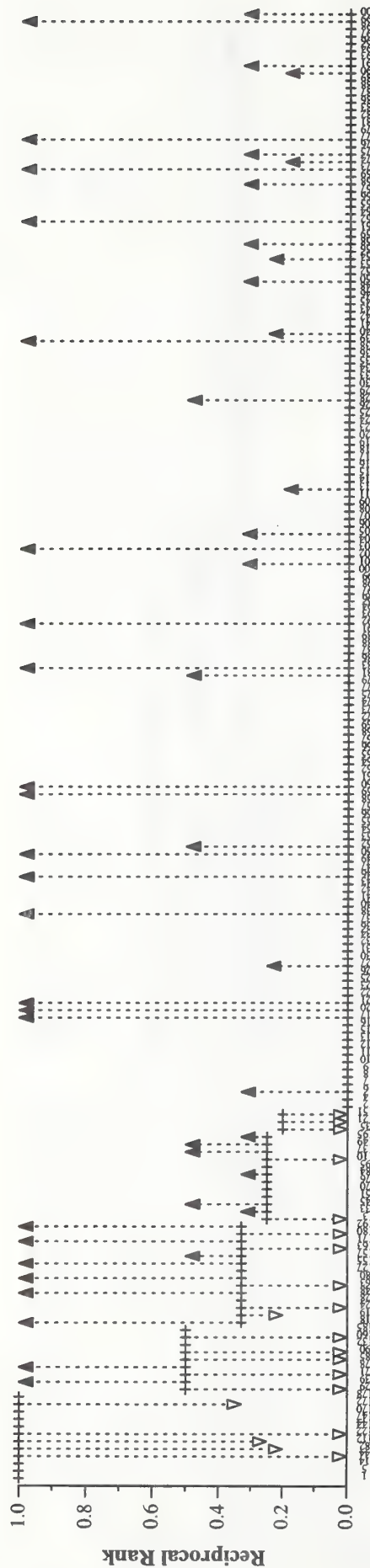
Run:	NTU99
Answer length:	250
Mean reciprocal rank:	0.087
# questions with no answer found:	173



Question number by descending median

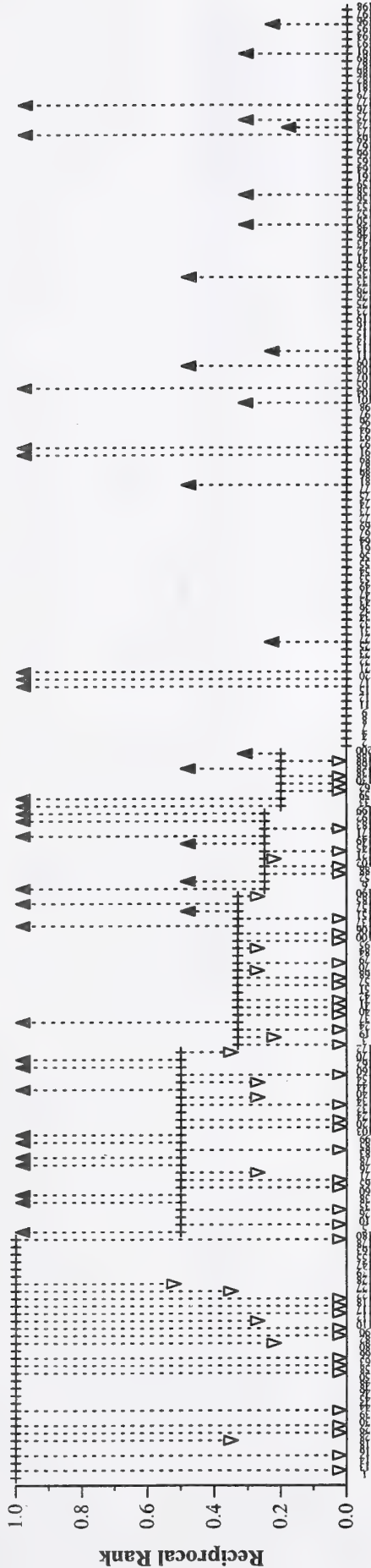
Run scores vs. median scores by question for run NTU99

Run:	CRL50
Answer length:	50
Mean reciprocal rank:	0.220
# questions with no answer found:	130



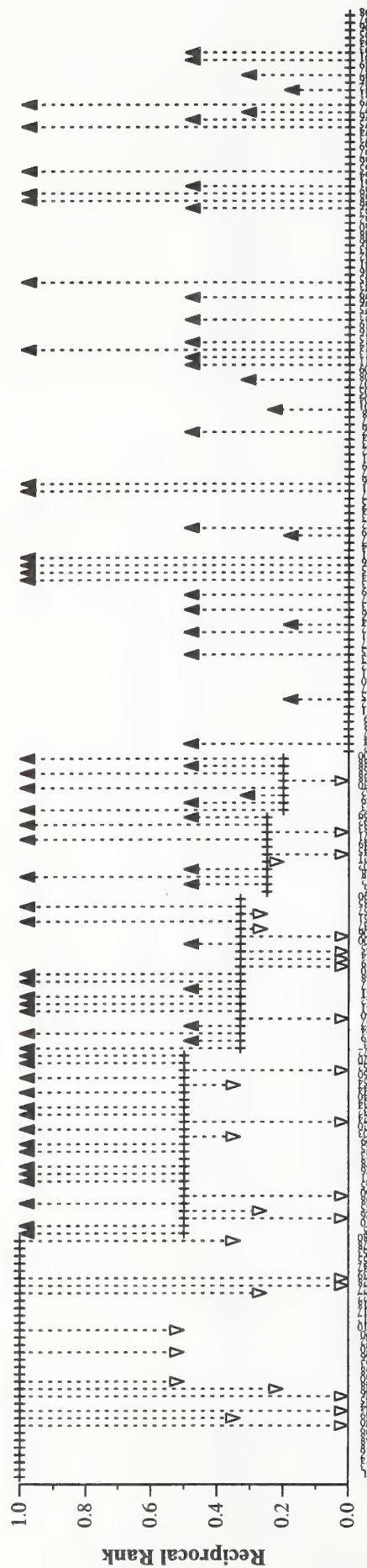
Question number by descending median
Run scores vs. median scores by question for run CRL50

Run:	CRL250
Answer length:	250
Mean reciprocal rank:	0.268
# questions with no answer found:	122



Run scores vs. median scores by question for run CRL250

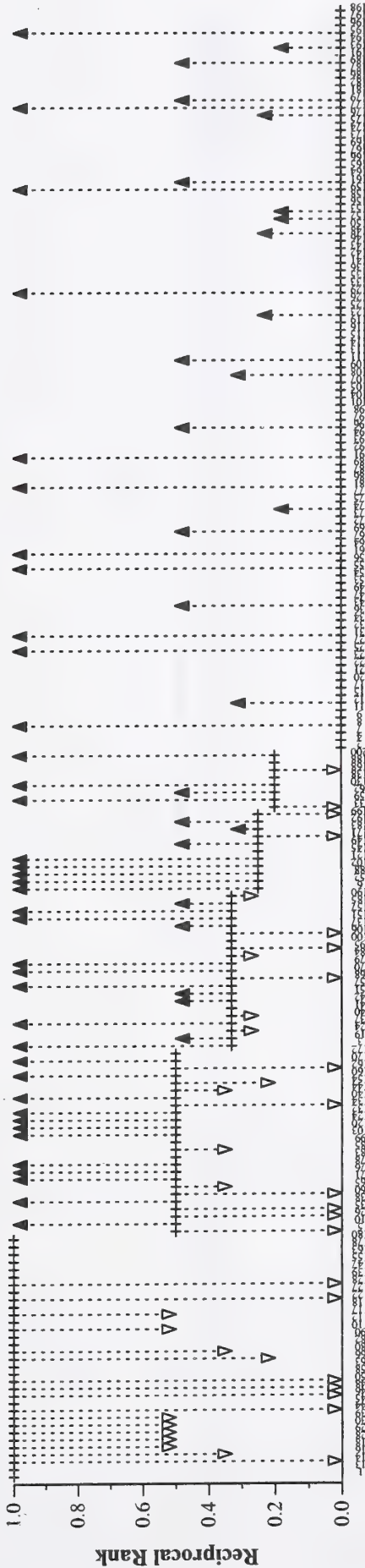
Run:	nttd8ql1
Answer length:	250
Mean reciprocal rank:	0.439
# questions with no answer found:	79



Question number by descending median

Run scores vs. median scores by question for run nttd8ql1

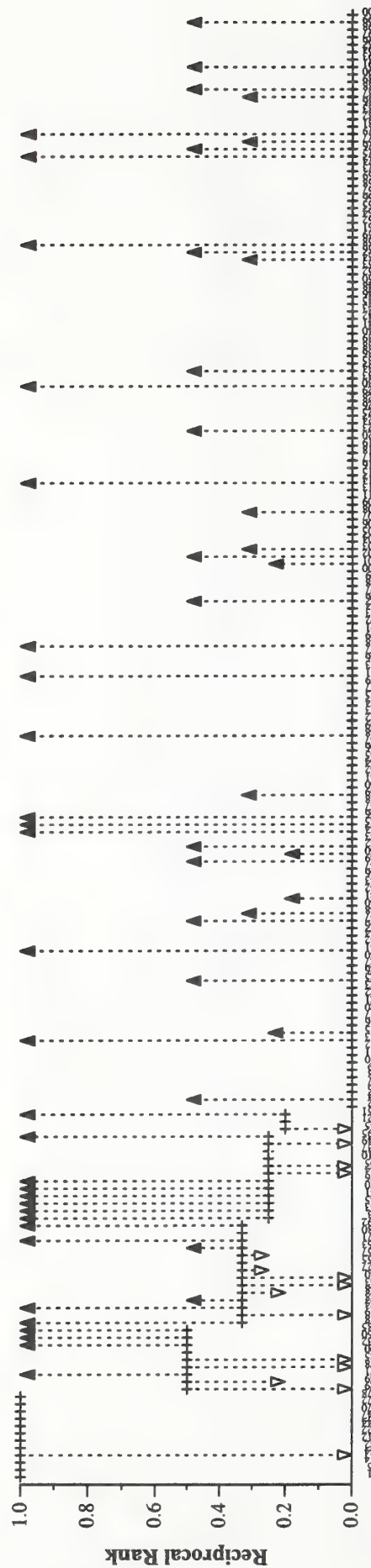
Run:	nttd8ql4
Answer length:	250
Mean reciprocal rank:	0.371
# questions with no answer found:	93



Question number by descending median

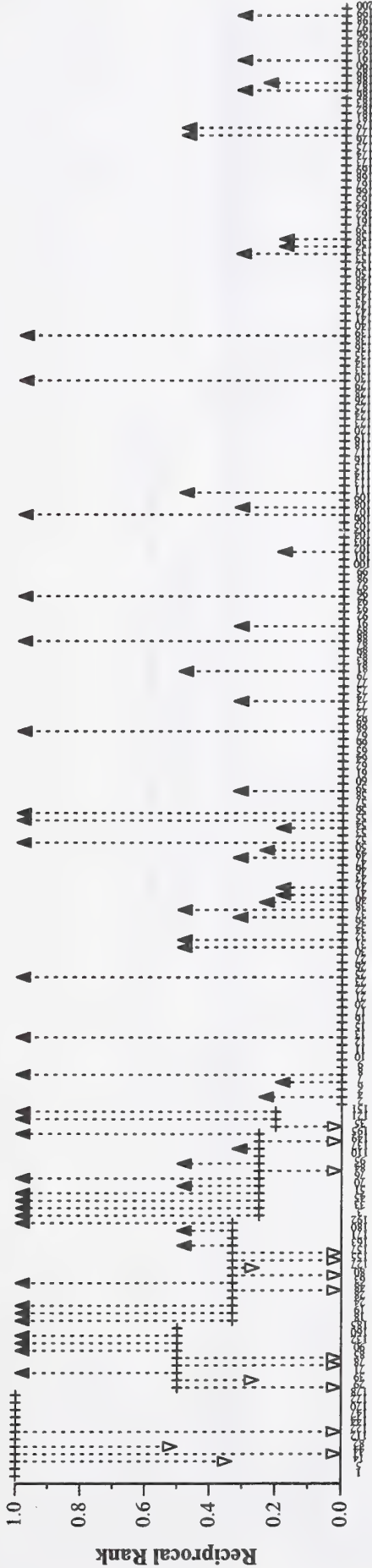
Run scores vs. median scores by question for run nttd8ql4

Run:	nttd8qs1
Answer length:	50
Mean reciprocal rank:	0.273
# questions with no answer found:	121



Question number by descending median
Run scores vs. median scores by question for run nttd8qs1

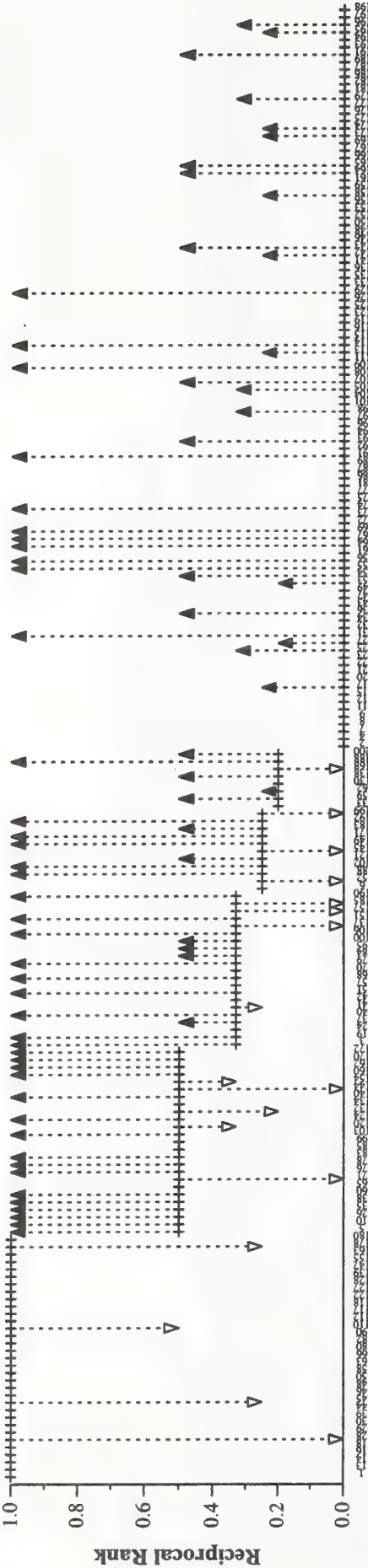
Run:	nttd8qs2
Answer length:	50
Mean reciprocal rank:	0.259
# questions with no answer found:	120



Question number by descending median

Run scores vs. median scores by question for run nttd8qs2

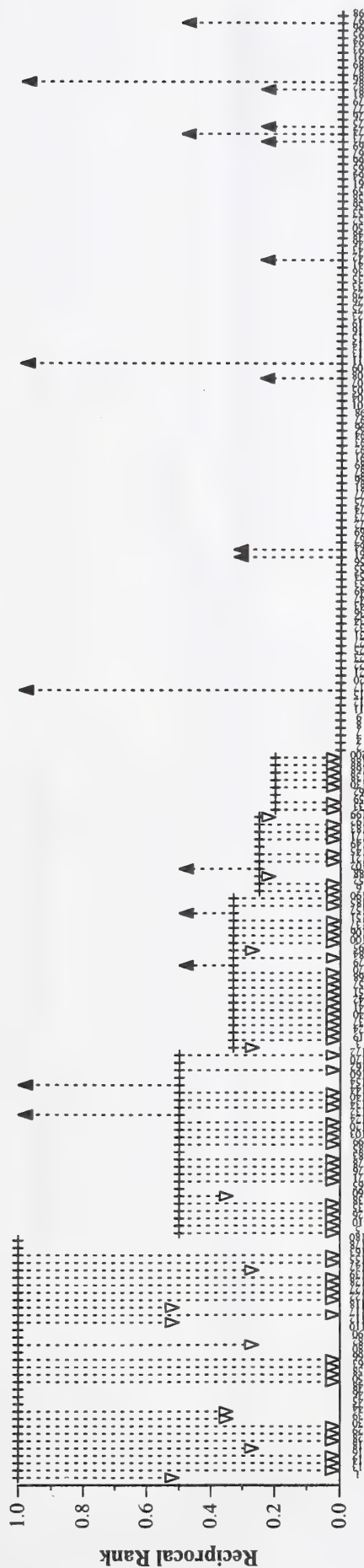
Run:	mds08q1
Answer length:	250
Mean reciprocal rank:	0.453
# questions with no answer found:	77



Question number by descending median

Run scores vs. median scores by question for run mds08q1

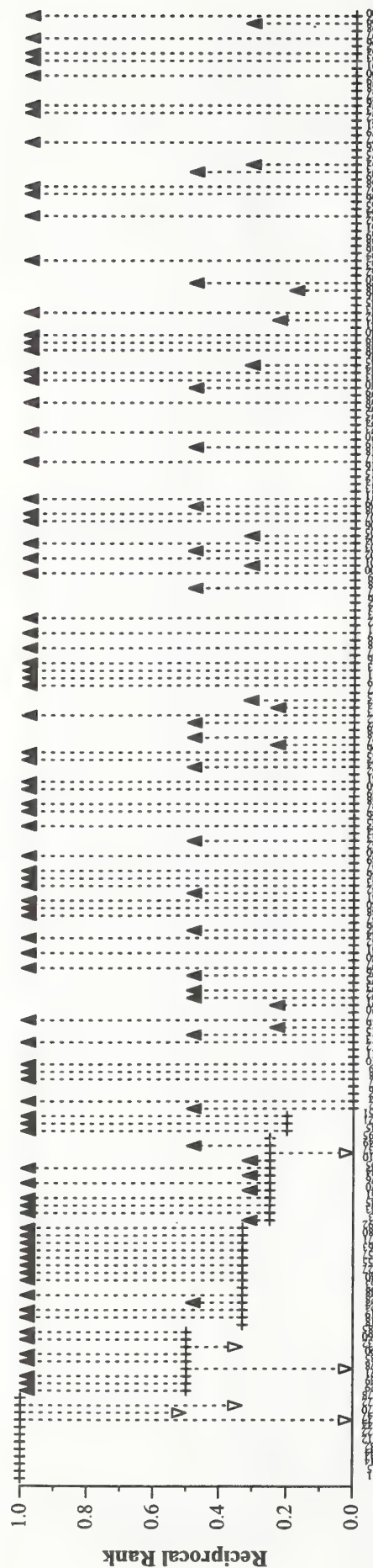
Run:	Scai8QnA
Answer length:	250
Mean reciprocal rank:	0.121
# questions with no answer found:	154



Question number by descending median

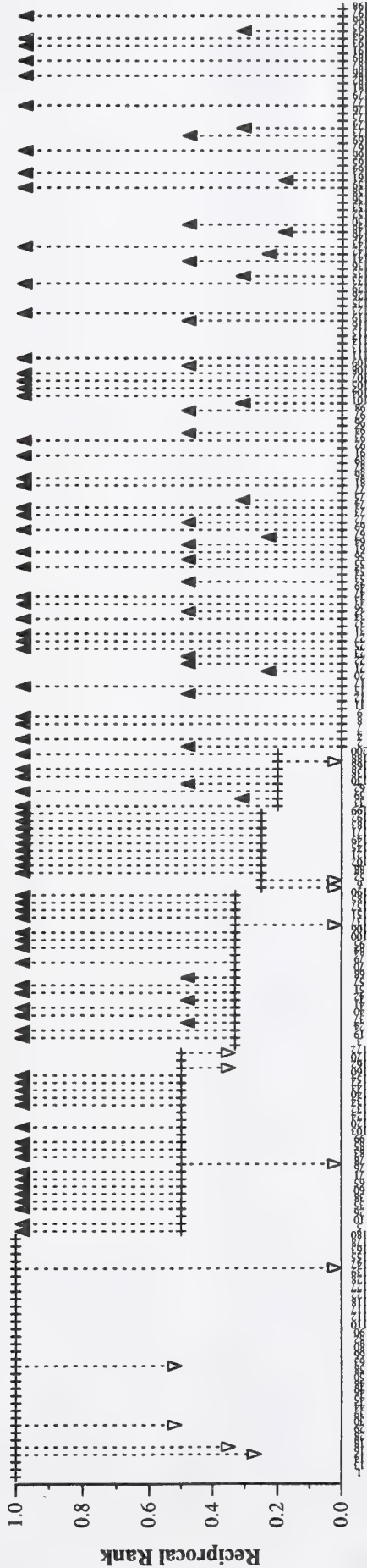
Run scores vs. median scores by question for run Scai8QnA

Run:	SMUNLP1
Answer length:	50
Mean reciprocal rank:	0.555
# questions with no answer found:	63



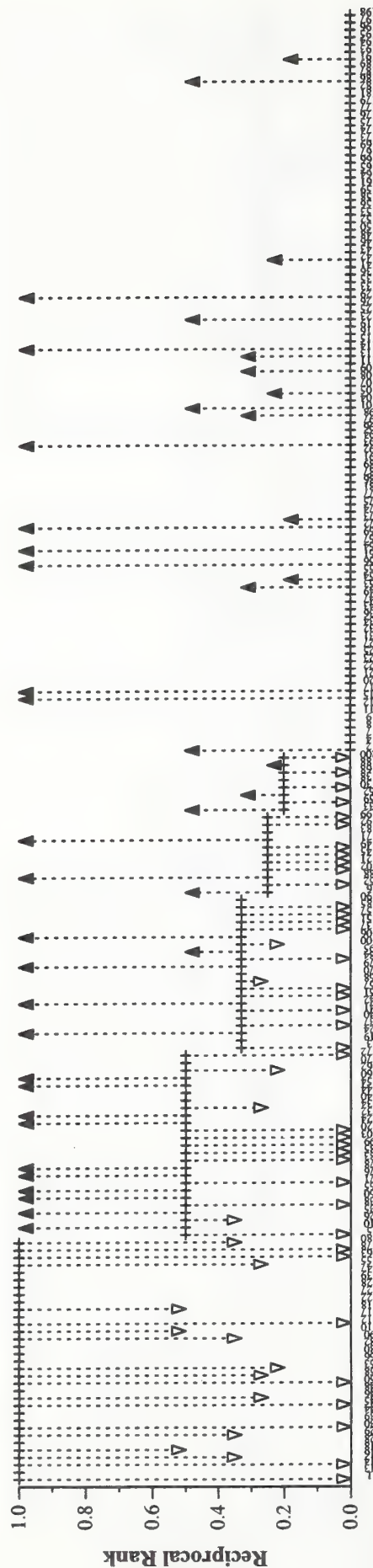
Question number by descending median
Run scores vs. median scores by question for run SMUNLP1

Run:	SMUNLP2
Answer length:	250
Mean reciprocal rank:	0.646
# questions with no answer found:	44



Question number by descending median
Run scores vs. median scores by question for run SMUNLP2

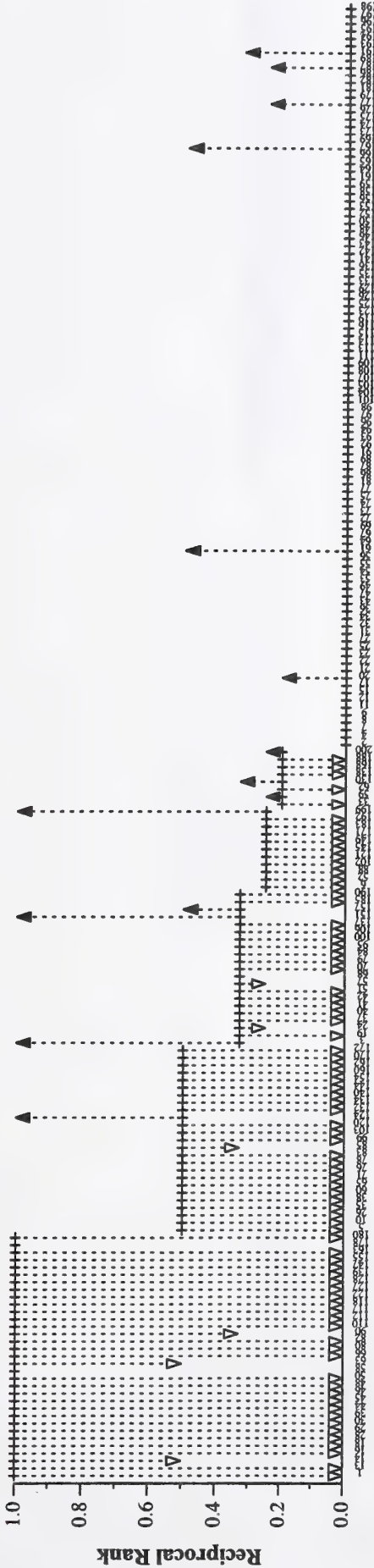
Run:	UIowaQA1
Answer length:	250
Mean reciprocal rank:	0.267
# questions with no answer found:	117



Question number by descending median

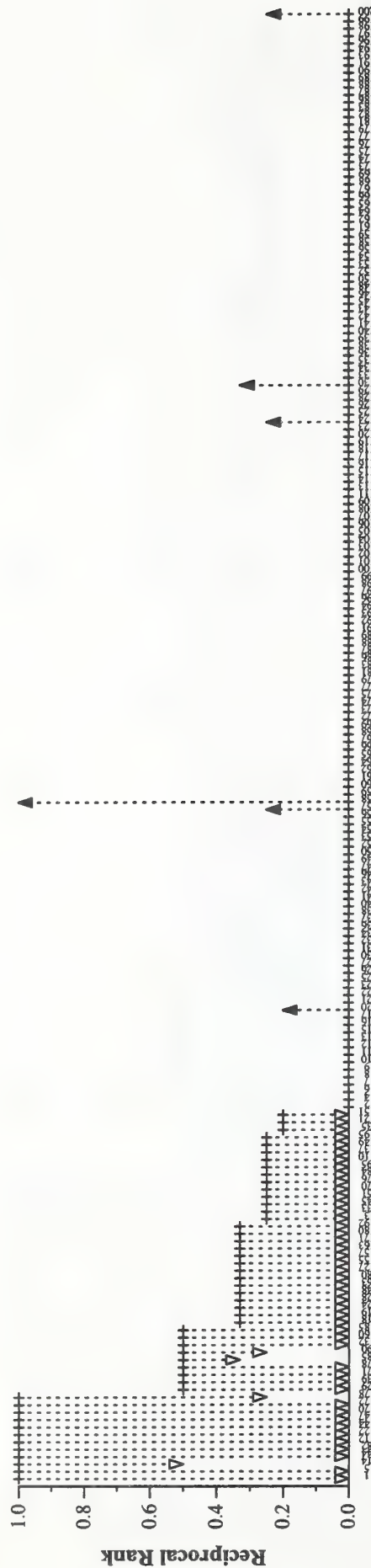
Run scores vs. median scores by question for run UIowaQA1

Run:	UIowaQA2
Answer length:	250
Mean reciprocal rank:	0.060
# questions with no answer found:	175



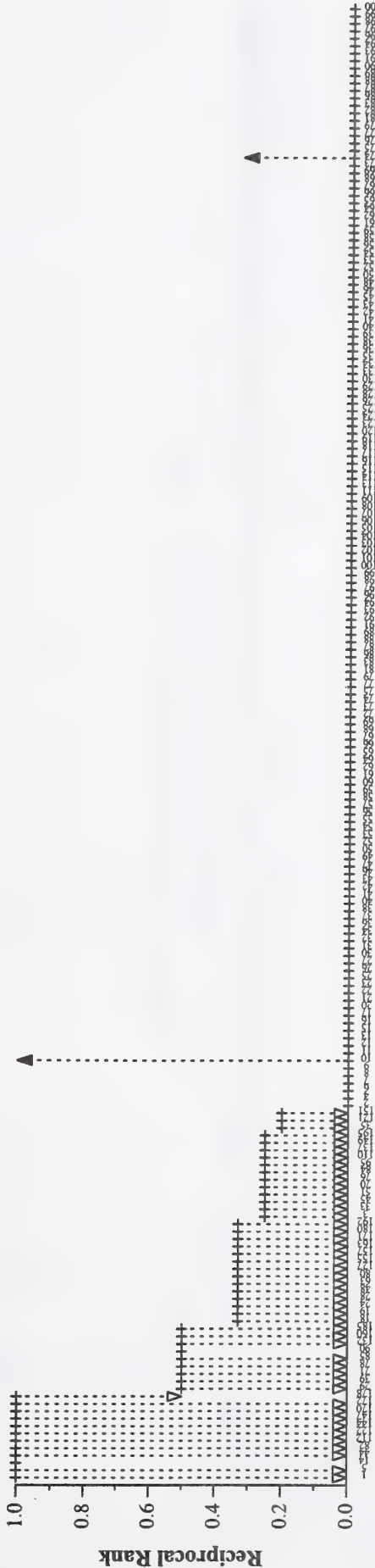
Question number by descending median
Run scores vs. median scores by question for run UIowaQA2

Run:	UIowaQA3
Answer length:	50
Mean reciprocal rank:	0.018
# questions with no answer found:	188



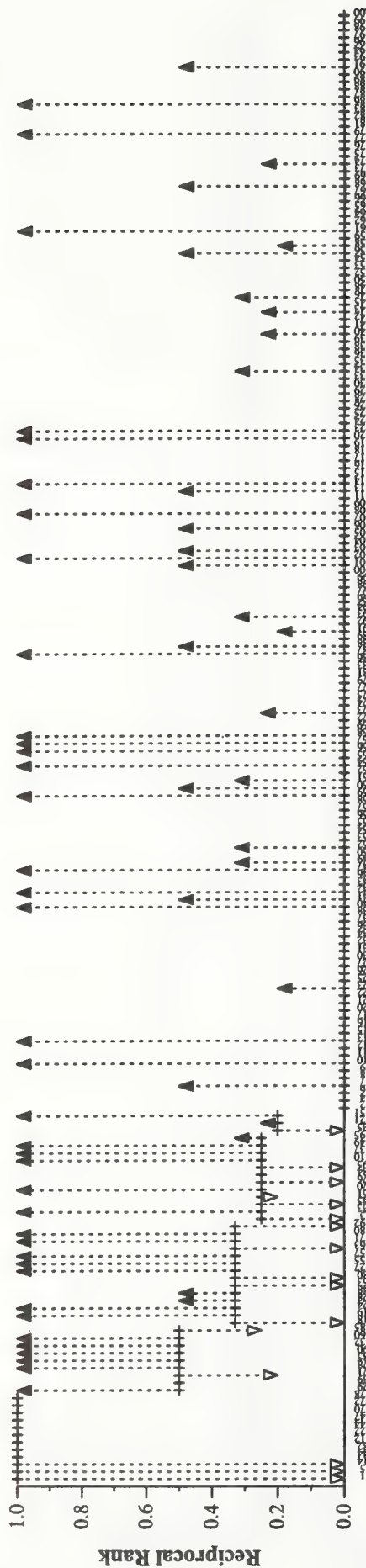
Question number by descending median
Run scores vs. median scores by question for run UIowaQA3

Run:	UIowaQA4
Answer length:	50
Mean reciprocal rank:	0.017
# questions with no answer found:	193



Question number by descending median
Run scores vs. median scores by question for run UIowaQA4

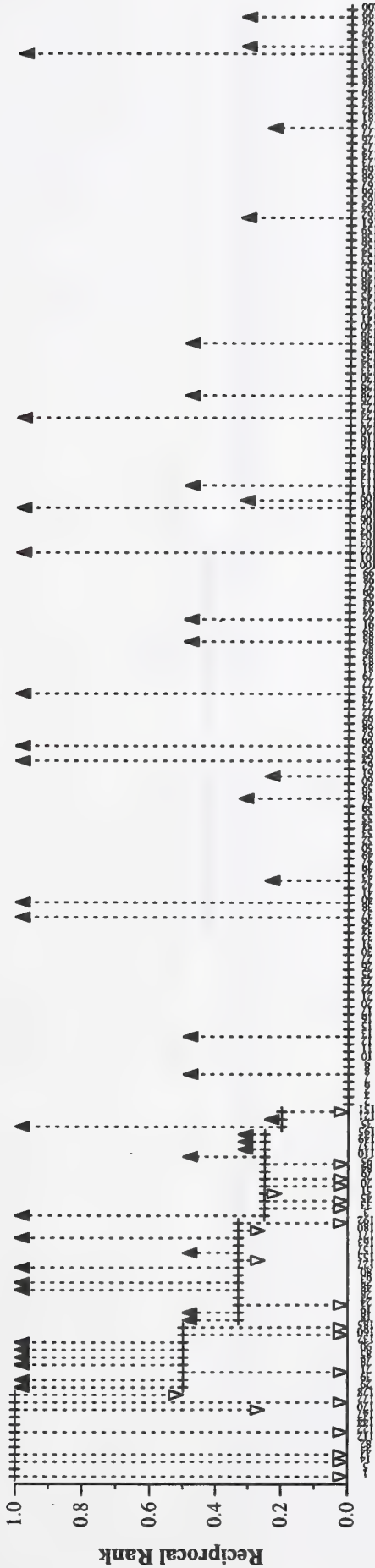
Run:	umdqa
Answer length:	50
Mean reciprocal rank:	0.298
# questions with no answer found:	118



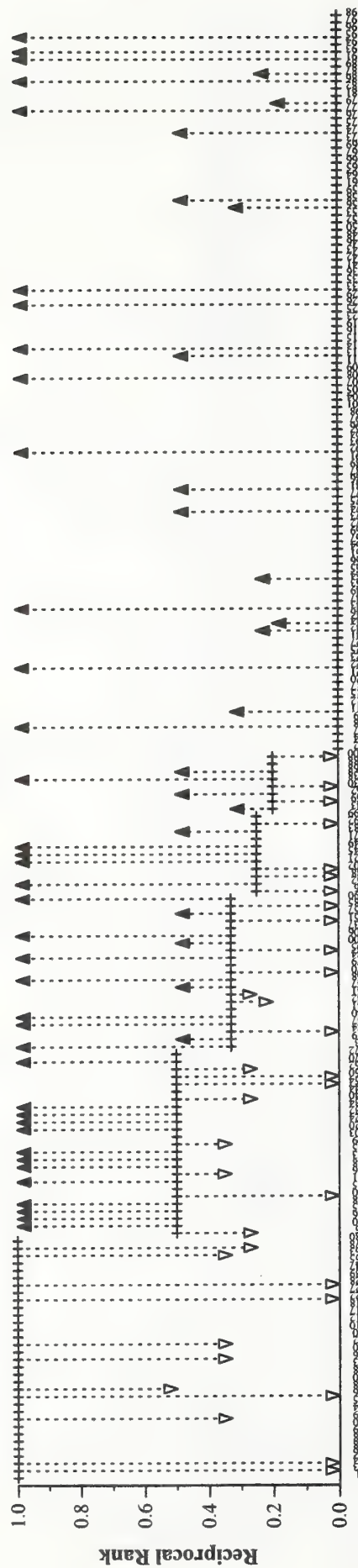
Question number by descending median

Run scores vs. median scores by question for run umdqa

Run:	INQ634
Answer length:	50
Mean reciprocal rank:	0.191
# questions with no answer found:	140



Run:	INQ635
Answer length:	250
Mean reciprocal rank:	0.383
# questions with no answer found:	95



Question number by descending median
Run scores vs. median scores by question for run INQ635

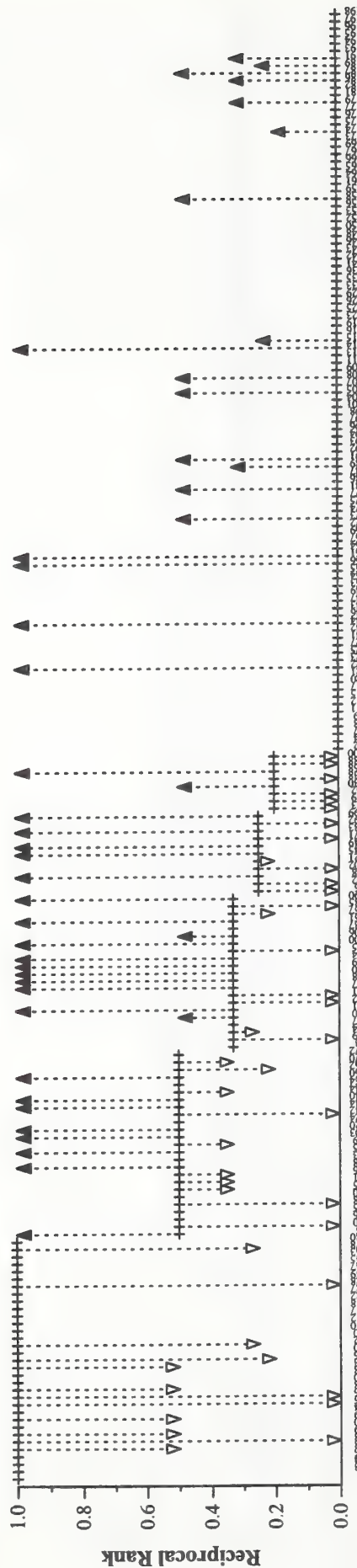
Run:	INQ638
Answer length:	50
Mean reciprocal rank:	0.126
# questions with no answer found:	158



Question number by descending median

Run scores vs. median scores by question for run INQ638

Run:	INQ639
Answer length:	250
Mean reciprocal rank:	0.336
# questions with no answer found:	104



Question number by descending median

Run scores vs. median scores by question for run INQ639

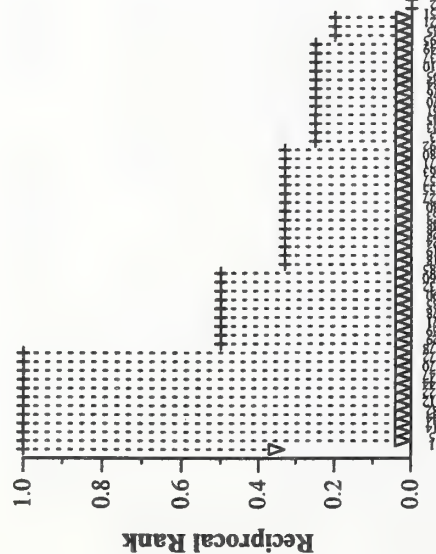
Run:	UOandNRC
Answer length:	250
Mean reciprocal rank:	0.002
# questions with no answer found:	197



Question number by descending median

Run scores vs. median scores by question for run UOandNRC

Run:	UOandNRC50
Answer length:	50
Mean reciprocal rank:	0.002
# questions with no answer found:	197



Question number by descending median

Run scores vs. median scores by question for run UOandNRC50

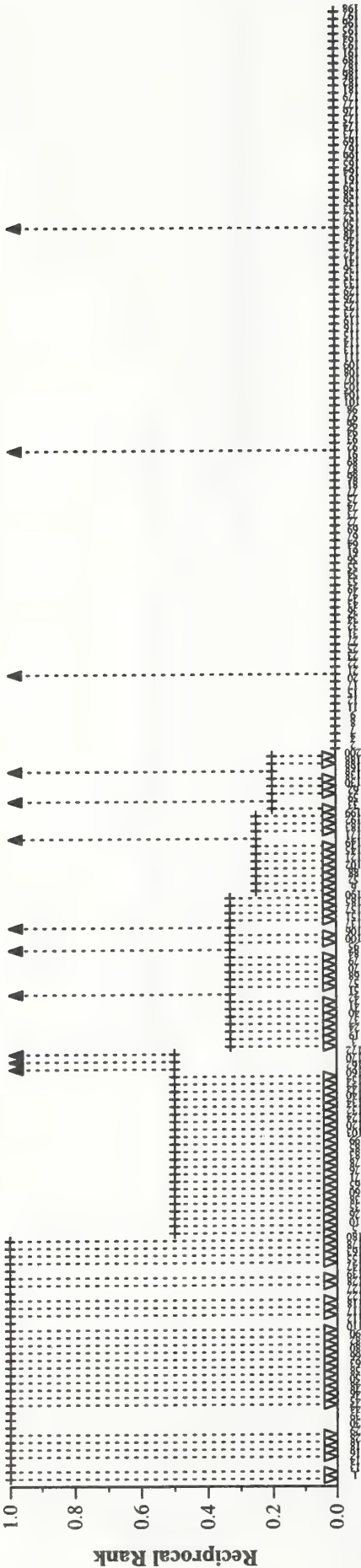
Run:	shefatt50
Answer length:	50
Mean reciprocal rank:	0.071
# questions with no answer found:	184



Question number by descending median

Run scores vs. median scores by question for run shefatt50

Run:	shefatt250
Answer length:	250
Mean reciprocal rank:	0.096
# questions with no answer found:	179



Question number by descending median

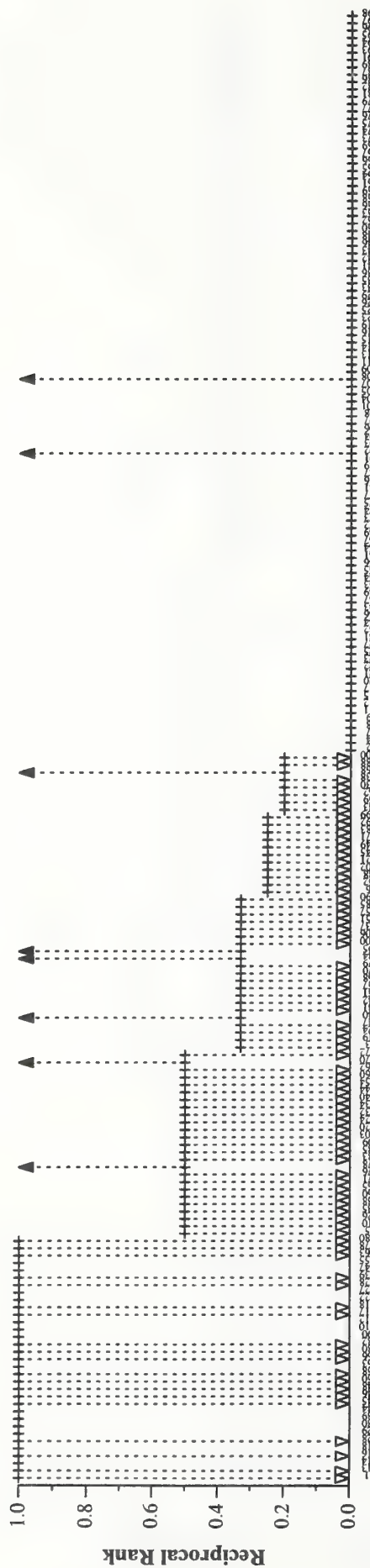
Run scores vs. median scores by question for run shefatt250

Run:	shefinq50
Answer length:	50
Mean reciprocal rank:	0.081
# questions with no answer found:	182



Run scores vs. median scores by question for run shefinq50

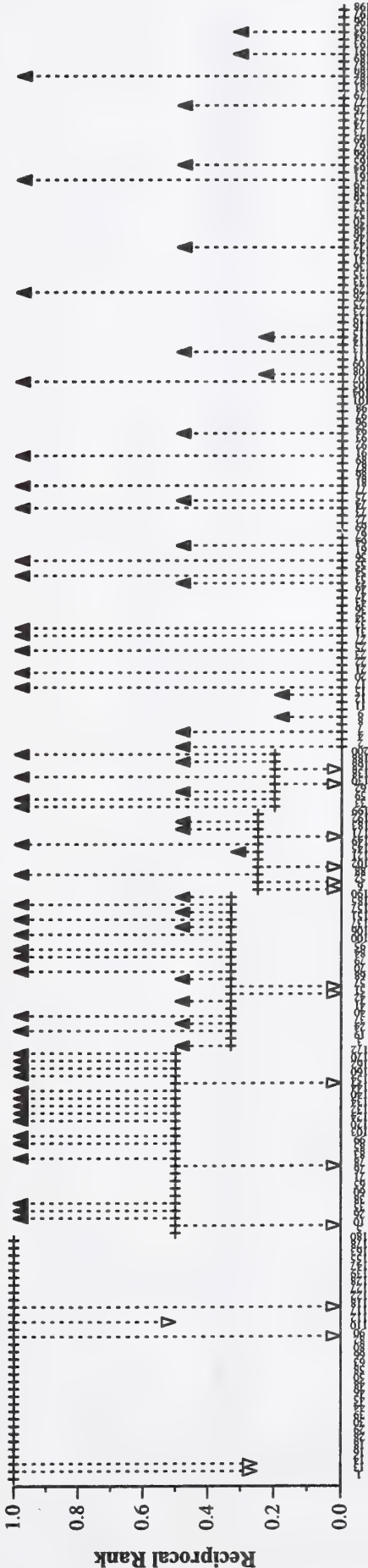
Run:	shefinq250
Answer length:	250
Mean reciprocal rank:	0.111
# questions with no answer found:	176



Question number by descending median

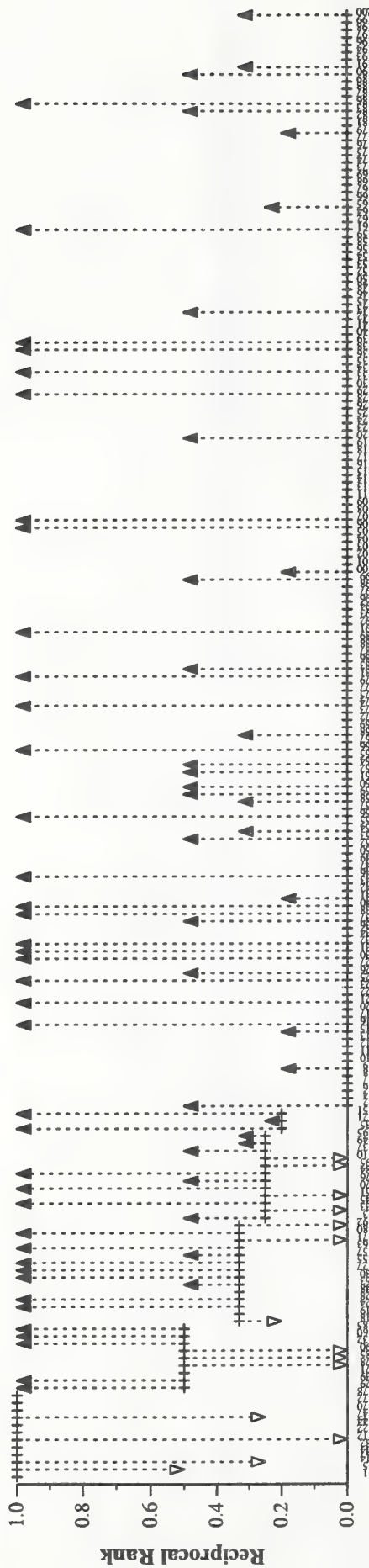
Run scores vs. median scores by question for run shefinq250

Run:	xeroxQA81C
Answer length:	250
Mean reciprocal rank:	0.453
# questions with no answer found:	83



Run scores vs. median scores by question for run xeroxQA81C

Run:	xeroxQA8sC
Answer length:	50
Mean reciprocal rank:	0.317
# questions with no answer found:	111

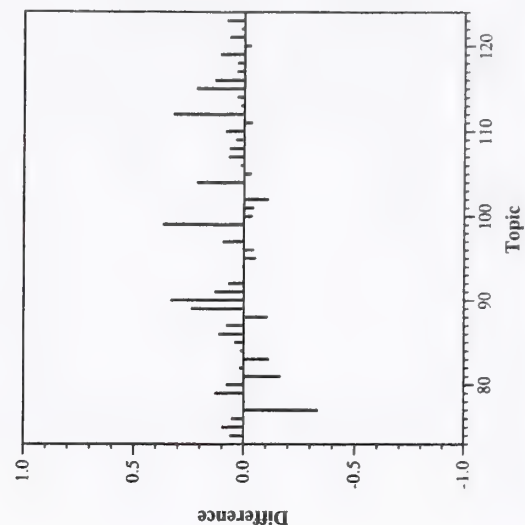
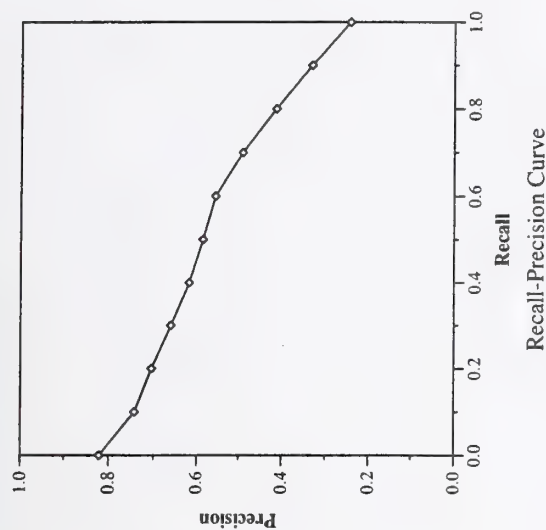


Question number by descending median
Run scores vs. median scores by question for run xeroxQA8sC

Summary Statistics		
Run Number	att-bl	
Run Description	known, rolling	
Number of Topics	49	
Total number of documents over all topics		
Retrieved:	49000	
Relevant:	1818	
Rel-ret:	1691	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8205
0.10	0.7429
0.20	0.7052
0.30	0.6609
0.40	0.6183
0.50	0.5860
0.60	0.5571
0.70	0.4917
0.80	0.4141
0.90	0.3319
1.00	0.2436
Average precision over all relevant docs	
non-interpolated	0.5539

Document Level Averages	
	Precision
At 5 docs	0.7020
At 10 docs	0.6102
At 15 docs	0.5524
At 20 docs	0.5122
At 30 docs	0.4286
At 100 docs	0.2231
At 200 docs	0.1424
At 500 docs	0.0648
At 1000 docs	0.0345
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5241

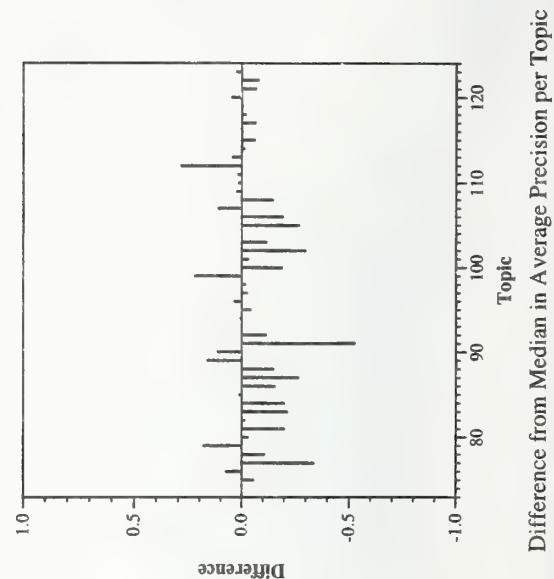
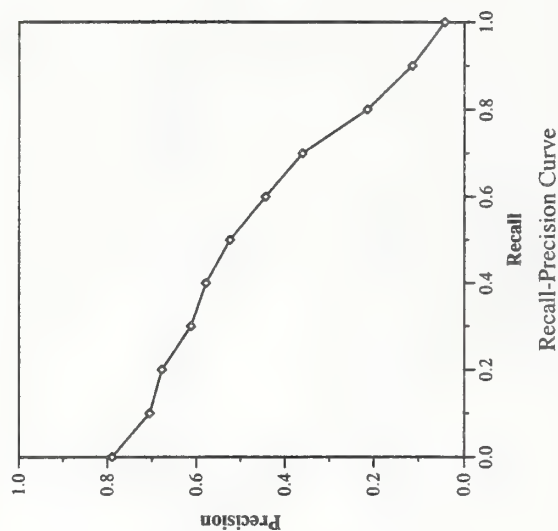


Spoken document retrieval track results — AT&T Labs Research

Summary Statistics	
Run Number	att-cr-cmus1
Run Description	known, rolling
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1479

Recall Level Precision Averages	
Recall	Precision
0.00	0.7895
0.10	0.7054
0.20	0.6782
0.30	0.6126
0.40	0.5787
0.50	0.5240
0.60	0.4438
0.70	0.3606
0.80	0.2146
0.90	0.1141
1.00	0.0426
Average precision over all relevant docs	
non-interpolated	0.4525

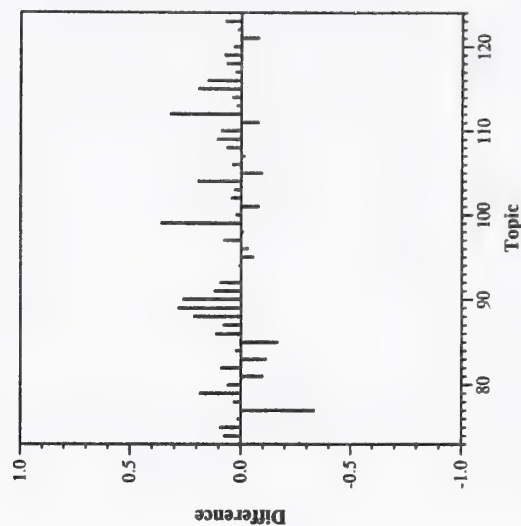
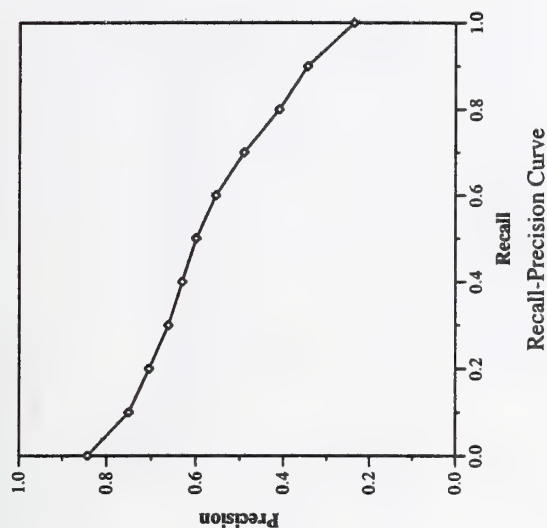
Document Level Averages	
	Precision
At 5 docs	0.6327
At 10 docs	0.5653
At 15 docs	0.5007
At 20 docs	0.4480
At 30 docs	0.3748
At 100 docs	0.1927
At 200 docs	0.1181
At 500 docs	0.0557
At 1000 docs	0.0302
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4639



Summary Statistics	
Run Number	att-cr-cuhtks1
Run Description	known, rolling
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1712

Recall Level Precision Averages	
Recall	Precision
0.00	0.8426
0.10	0.7506
0.20	0.7055
0.30	0.6621
0.40	0.6311
0.50	0.5995
0.60	0.5553
0.70	0.4906
0.80	0.4100
0.90	0.3451
1.00	0.2371
Average precision over all relevant docs	
non-interpolated	0.5592

Document Level Averages	
	Precision
At 5 docs	0.7143
At 10 docs	0.6061
At 15 docs	0.5442
At 20 docs	0.5051
At 30 docs	0.4299
At 100 docs	0.2278
At 200 docs	0.1464
At 500 docs	0.0669
At 1000 docs	0.0349
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5228

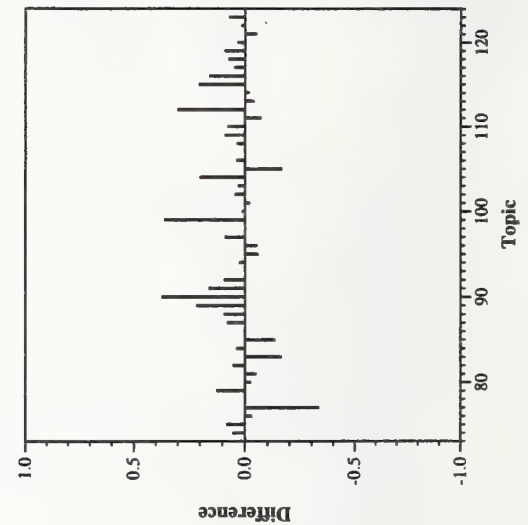
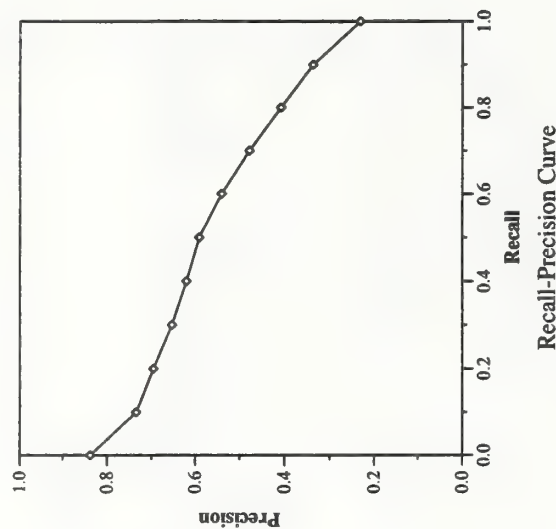


Spoken document retrieval track results — AT&T Labs Research

Summary Statistics	
Run Number	att-cr-cuhtks1p1
Run Description	known, rolling
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1708

Recall Level Precision Averages	
Recall	Precision
0.00	0.8375
0.10	0.7344
0.20	0.6958
0.30	0.6536
0.40	0.6203
0.50	0.5912
0.60	0.5402
0.70	0.4782
0.80	0.4084
0.90	0.3374
1.00	0.2325
Average precision over all relevant docs	
non-interpolated	0.5494

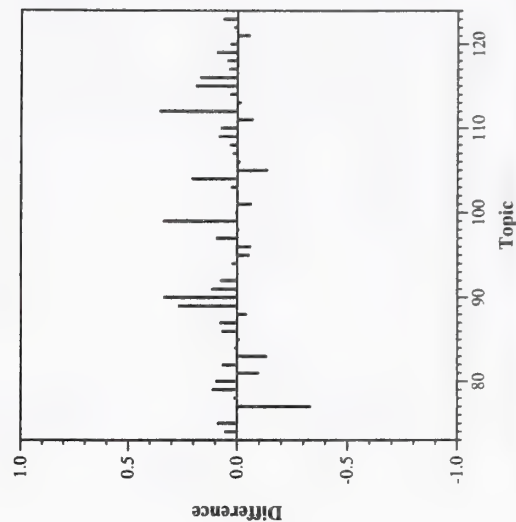
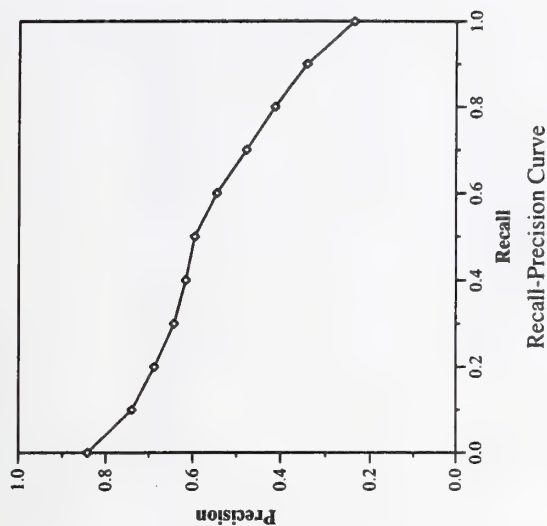
Document Level Averages	
	Precision
At 5 docs	0.6816
At 10 docs	0.6082
At 15 docs	0.5442
At 20 docs	0.5082
At 30 docs	0.4354
At 100 docs	0.2259
At 200 docs	0.1450
At 500 docs	0.0666
At 1000 docs	0.0349
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5171



Summary Statistics	
Run Number	att-cr-limsis1
Run Description	known, rolling
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1705

Recall Level Precision Averages	
Recall	Precision
0.00	0.8421
0.10	0.7396
0.20	0.6879
0.30	0.6426
0.40	0.6169
0.50	0.5964
0.60	0.5462
0.70	0.4783
0.80	0.4138
0.90	0.3424
1.00	0.2376
Average precision over all relevant docs	
non-interpolated	0.5517

Document Level Averages	
At 5 docs	0.6939
At 10 docs	0.6061
At 15 docs	0.5524
At 20 docs	0.4959
At 30 docs	0.4333
At 100 docs	0.2259
At 200 docs	0.1447
At 500 docs	0.0664
At 1000 docs	0.0348
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5165

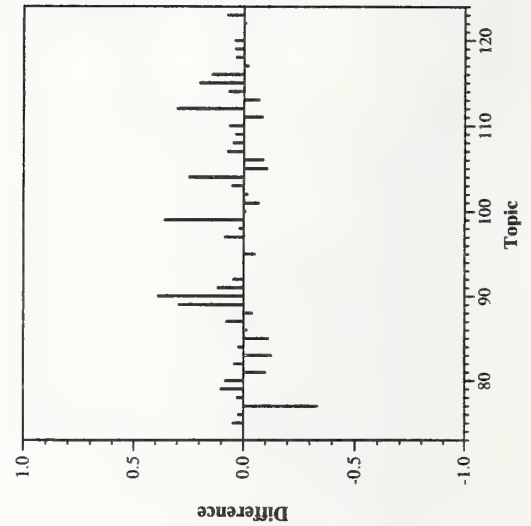
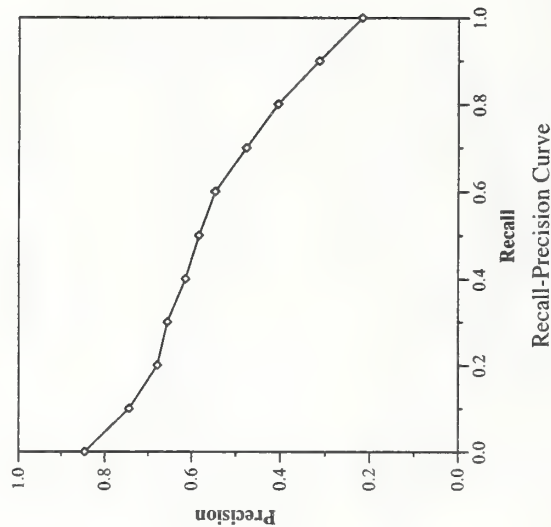


Spoken document retrieval track results — AT&T Labs Research

Summary Statistics	
Run Number	att-cr-shesl
Run Description	known, rolling
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1694

Recall Level Precision Averages	
Recall	Precision
0.00	0.8457
0.10	0.7441
0.20	0.6791
0.30	0.6556
0.40	0.6141
0.50	0.5833
0.60	0.5469
0.70	0.4765
0.80	0.4054
0.90	0.3139
1.00	0.2178
Average precision over all relevant docs	
non-interpolated	0.5455

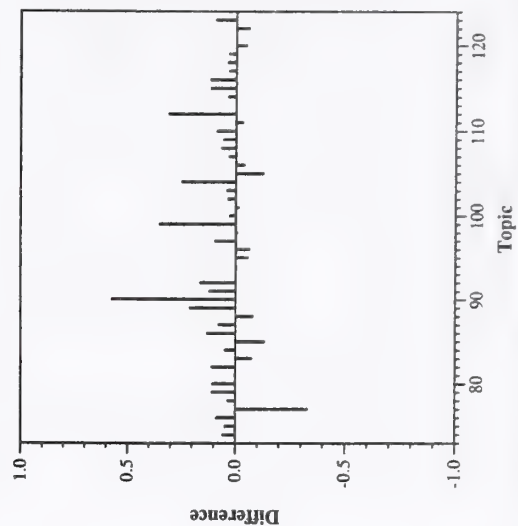
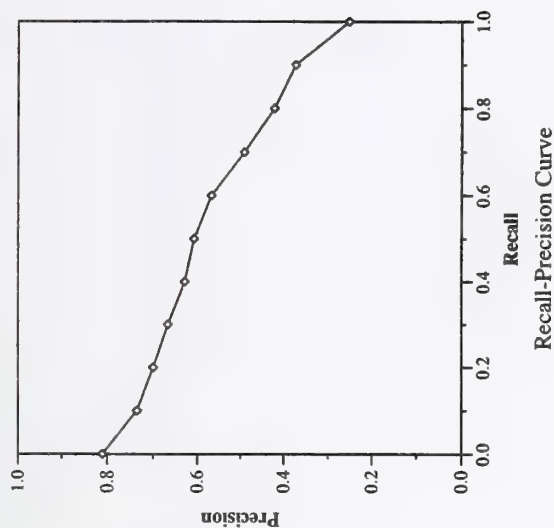
Document Level Averages	
	Precision
At 5 docs	0.6816
At 10 docs	0.6163
At 15 docs	0.5483
At 20 docs	0.4929
At 30 docs	0.4218
At 100 docs	0.2251
At 200 docs	0.1442
At 500 docs	0.0658
At 1000 docs	0.0346
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5167



Summary Statistics		
Run Number	att-r1	
Run Description	known, rolling	
Number of Topics	49	
Total number of documents over all topics		
Retrieved:	49000	
Relevant:	1818	
Rel-ret:	1724	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8106
0.10	0.7355
0.20	0.6999
0.30	0.6664
0.40	0.6287
0.50	0.6073
0.60	0.5678
0.70	0.4909
0.80	0.4211
0.90	0.3739
1.00	0.2527
Average precision over all relevant docs	
non-interpolated	0.5598

Document Level Averages	
	Precision
At 5 docs	0.6857
At 10 docs	0.6041
At 15 docs	0.5469
At 20 docs	0.5000
At 30 docs	0.4313
At 100 docs	0.2278
At 200 docs	0.1453
At 500 docs	0.0671
At 1000 docs	0.0352
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5320

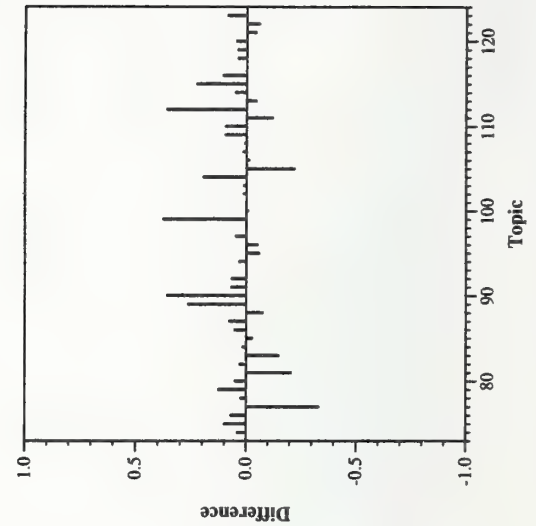
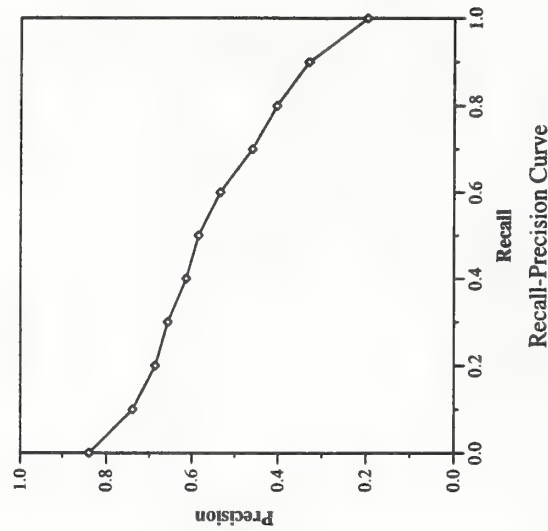


Spoken document retrieval track results — AT&T Labs Research

Summary Statistics		
Run Number	att-sl	
Run Description	known, rolling	
Number of Topics	49	
Total number of documents over all topics		
Retrieved:	49000	
Relevant:	1818	
Rel-ret:	1695	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8390
0.10	0.7390
0.20	0.6861
0.30	0.6570
0.40	0.6135
0.50	0.5842
0.60	0.5340
0.70	0.4599
0.80	0.4036
0.90	0.3299
1.00	0.1967
Average precision over all relevant docs	
non-interpolated	0.5431

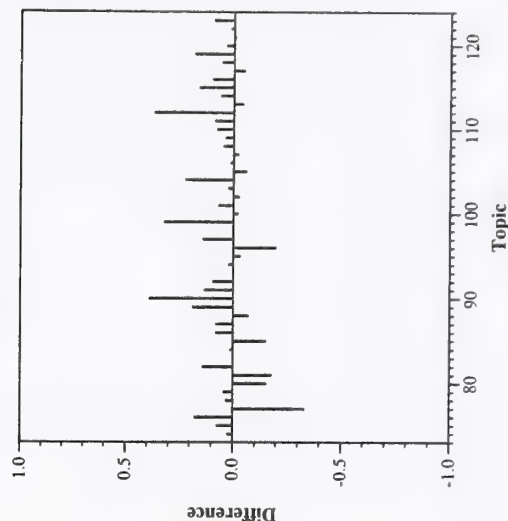
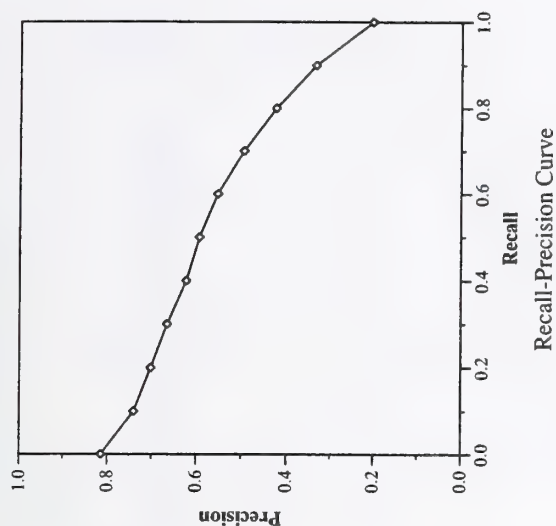
Document Level Averages	
	Precision
At 5 docs	0.6898
At 10 docs	0.6020
At 15 docs	0.5388
At 20 docs	0.4898
At 30 docs	0.4190
At 100 docs	0.2257
At 200 docs	0.1434
At 500 docs	0.0663
At 1000 docs	0.0346
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5217



Summary Statistics		
Run Number	att-s2	
Run Description	known, rolling	
Number of Topics	49	
Total number of documents over all topics		
Retrieved:	49000	
Relevant:	1818	
Rel-ret:	1678	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8141
0.10	0.7403
0.20	0.7021
0.30	0.6653
0.40	0.6229
0.50	0.5935
0.60	0.5534
0.70	0.4953
0.80	0.4250
0.90	0.3349
1.00	0.2032
Average precision over all relevant docs	
non-interpolated	0.5510

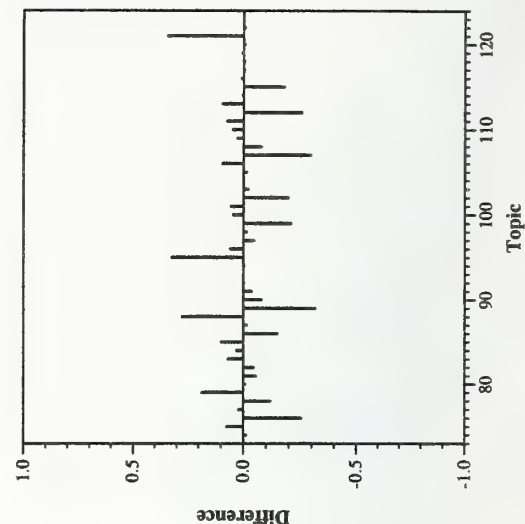
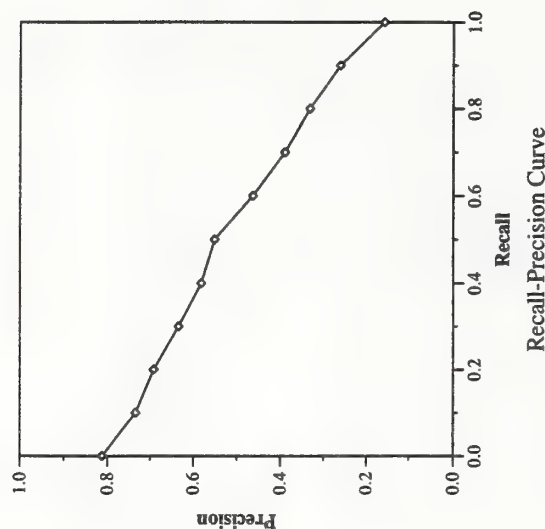
Document Level Averages	
	Precision
At 5 docs	0.6776
At 10 docs	0.6122
At 15 docs	0.5551
At 20 docs	0.5041
At 30 docs	0.4354
At 100 docs	0.2288
At 200 docs	0.1424
At 500 docs	0.0647
At 1000 docs	0.0342
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5326



Summary Statistics	
Run Number	cuhtk-b1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1552

Recall Level Precision Averages	
Recall	Precision
0.00	0.8122
0.10	0.7346
0.20	0.6921
0.30	0.6339
0.40	0.5822
0.50	0.5517
0.60	0.4636
0.70	0.3894
0.80	0.3318
0.90	0.2608
1.00	0.1577
Average precision over all relevant docs	
non-interpolated	0.4963

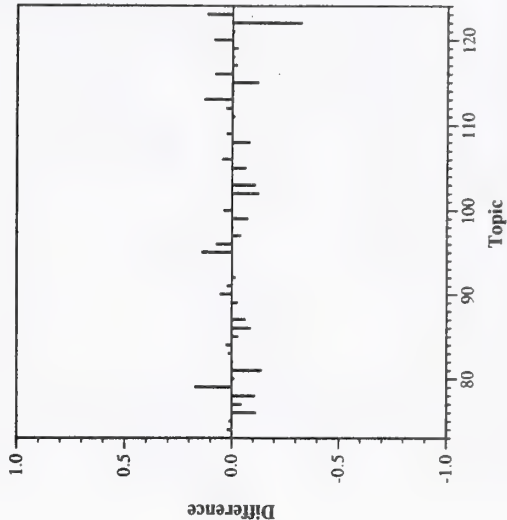
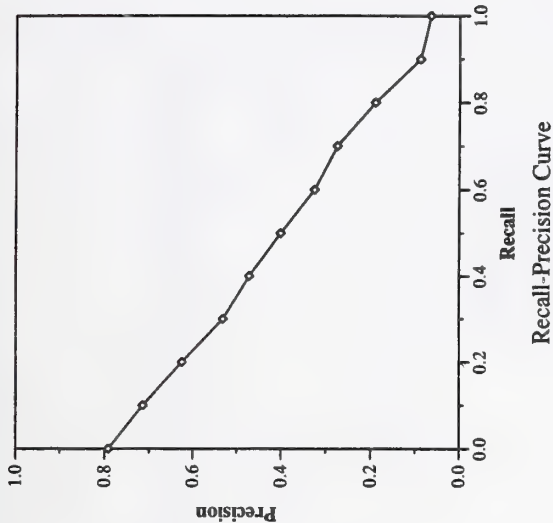
Document Level Averages	
	Precision
At 5 docs	0.6571
At 10 docs	0.5694
At 15 docs	0.5156
At 20 docs	0.4541
At 30 docs	0.3735
At 100 docs	0.1937
At 200 docs	0.1236
At 500 docs	0.0592
At 1000 docs	0.0317
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4638



Summary Statistics		
Run Number	cuhtk-b1u	
Run Description	unknown, fixed	
Number of Topics	49	
Total number of documents over all topics		
Retrieved:	49000	
Relevant:	1818	
Rel-ret:	1339	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7915
0.10	0.7147
0.20	0.6250
0.30	0.5314
0.40	0.4720
0.50	0.4010
0.60	0.3254
0.70	0.2744
0.80	0.1899
0.90	0.0889
1.00	0.0656
Average precision over all relevant docs	
non-interpolated	0.3862

Document Level Averages	
	Precision
At 5 docs	0.5878
At 10 docs	0.5082
At 15 docs	0.4395
At 20 docs	0.3888
At 30 docs	0.3197
At 100 docs	0.1692
At 200 docs	0.1064
At 500 docs	0.0509
At 1000 docs	0.0273
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4022

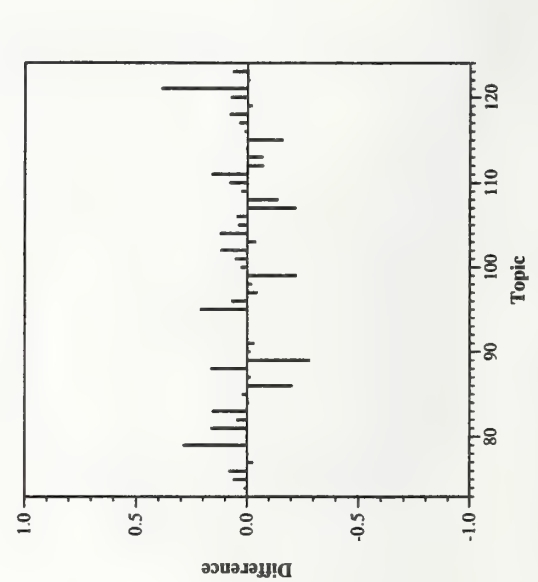
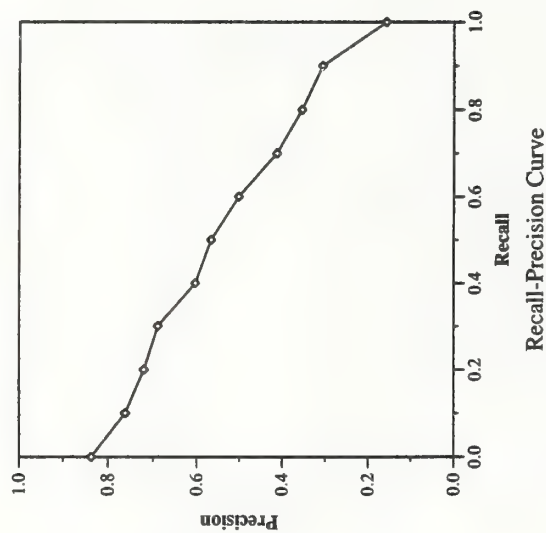


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	cuhtk-cr-att-sl
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1617

Recall Level Precision Averages	
Recall	Precision
0.00	0.8371
0.10	0.7623
0.20	0.7205
0.30	0.6895
0.40	0.6034
0.50	0.5667
0.60	0.5016
0.70	0.4126
0.80	0.3538
0.90	0.3062
1.00	0.1579
Average precision over all relevant docs	
non-interpolated	0.5275

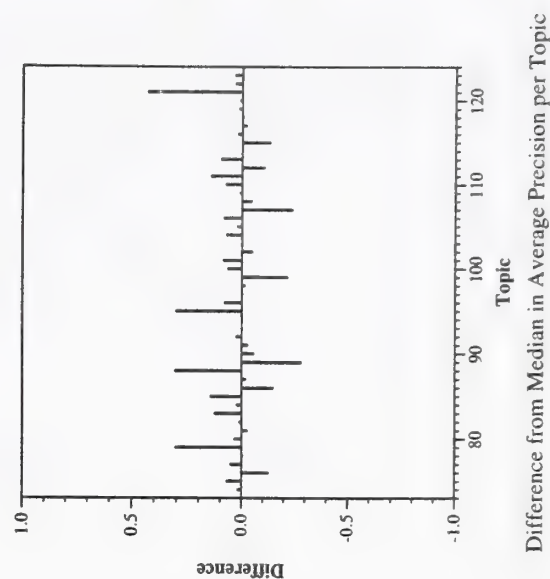
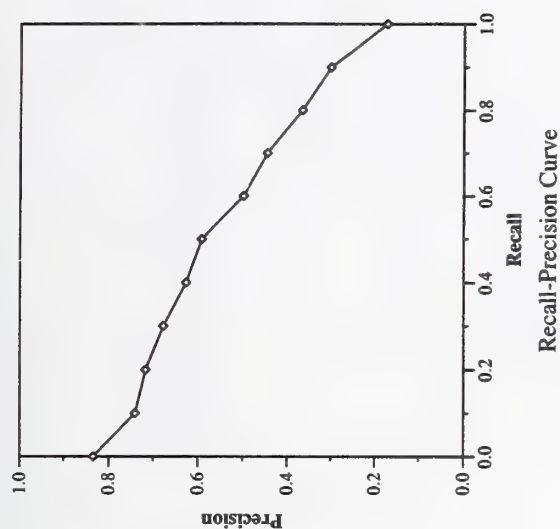
Document Level Averages	
	Precision
At 5 docs	0.6857
At 10 docs	0.5918
At 15 docs	0.5279
At 20 docs	0.4694
At 30 docs	0.3898
At 100 docs	0.2004
At 200 docs	0.1329
At 500 docs	0.0633
At 1000 docs	0.0330
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5107



Summary Statistics	
Run Number	cuhtk-cr-b2
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1599

Recall Level Precision Averages	
Recall	Precision
0.00	0.8341
0.10	0.7409
0.20	0.7173
0.30	0.6773
0.40	0.6265
0.50	0.5914
0.60	0.4988
0.70	0.4455
0.80	0.3667
0.90	0.3012
1.00	0.1729
Average precision over all relevant docs	
non-interpolated	0.5302

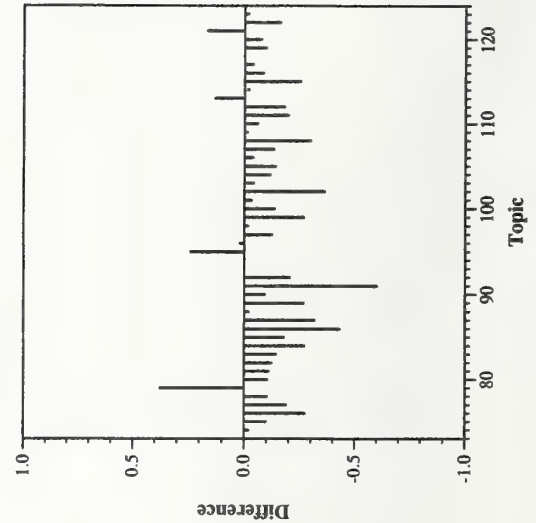
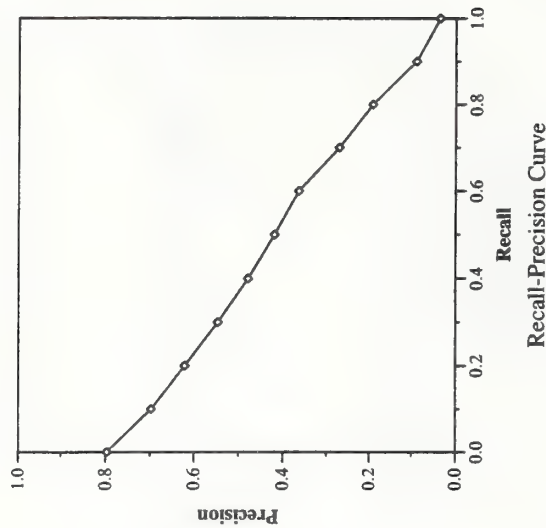
Document Level Averages	
	Precision
At 5 docs	0.6857
At 10 docs	0.5837
At 15 docs	0.5361
At 20 docs	0.4816
At 30 docs	0.3946
At 100 docs	0.2014
At 200 docs	0.1299
At 500 docs	0.0616
At 1000 docs	0.0326
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4989



Summary Statistics	
Run Number	cuhtk-cr-cmu-sl
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1368

Recall Level Precision Averages	
Recall	Precision
0.00	0.7970
0.10	0.6974
0.20	0.6210
0.30	0.5468
0.40	0.4790
0.50	0.4193
0.60	0.3643
0.70	0.2706
0.80	0.1920
0.90	0.0897
1.00	0.0360
Average precision over all relevant docs	
non-interpolated	0.3936

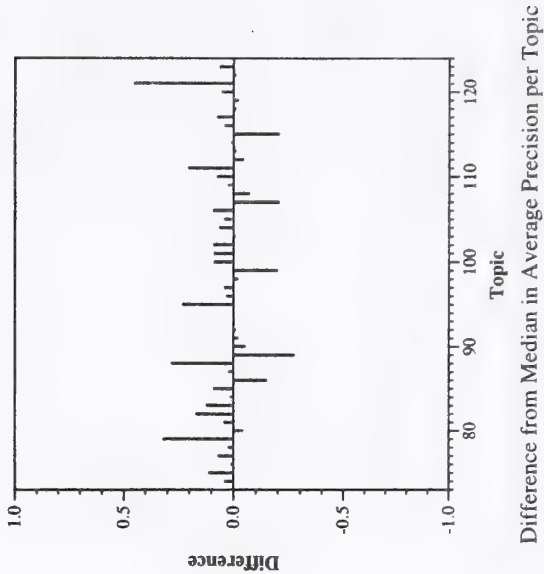
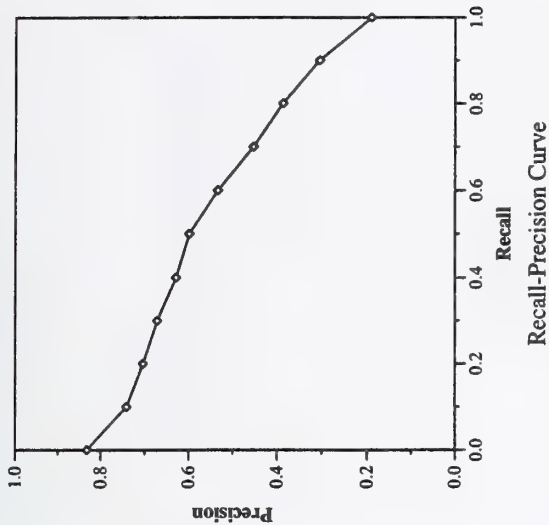
Document Level Averages	
	Precision
At 5 docs	0.6041
At 10 docs	0.5082
At 15 docs	0.4476
At 20 docs	0.3939
At 30 docs	0.3245
At 100 docs	0.1614
At 200 docs	0.1073
At 500 docs	0.0513
At 1000 docs	0.0279
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4068



Summary Statistics	
Run Number	cuhtk-cr-limsi-s1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1637

Recall Level Precision Averages	
Recall	Precision
0.00	0.8314
0.10	0.7413
0.20	0.7042
0.30	0.6717
0.40	0.6292
0.50	0.5988
0.60	0.5338
0.70	0.4535
0.80	0.3867
0.90	0.3052
1.00	0.1899
Average precision over all relevant docs	
non-interpolated	0.5412

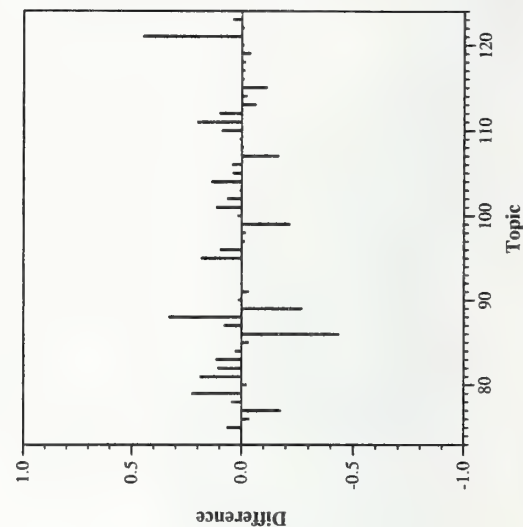
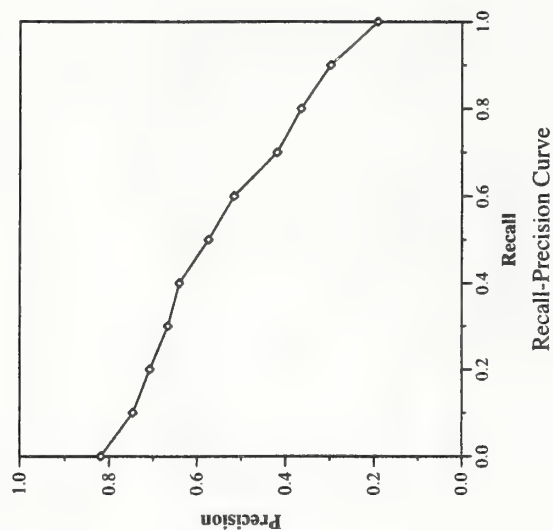
Document Level Averages	
	Precision
At 5 docs	0.7061
At 10 docs	0.6041
At 15 docs	0.5374
At 20 docs	0.4755
At 30 docs	0.3993
At 100 docs	0.2057
At 200 docs	0.1344
At 500 docs	0.0633
At 1000 docs	0.0334
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5026



Summary Statistics	
Run Number	cuhtk-cr-shef-s1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1604

Recall Level Precision Averages	
Recall	Precision
0.00	0.8192
0.10	0.7468
0.20	0.7080
0.30	0.6668
0.40	0.6405
0.50	0.5741
0.60	0.5167
0.70	0.4193
0.80	0.3650
0.90	0.2984
1.00	0.1914
Average precision over all relevant docs	
non-interpolated	0.5285

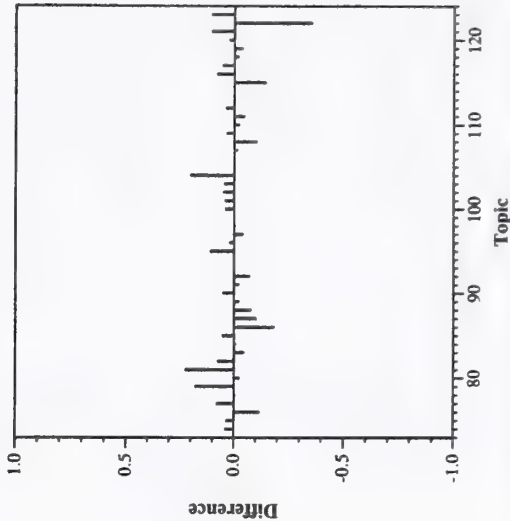
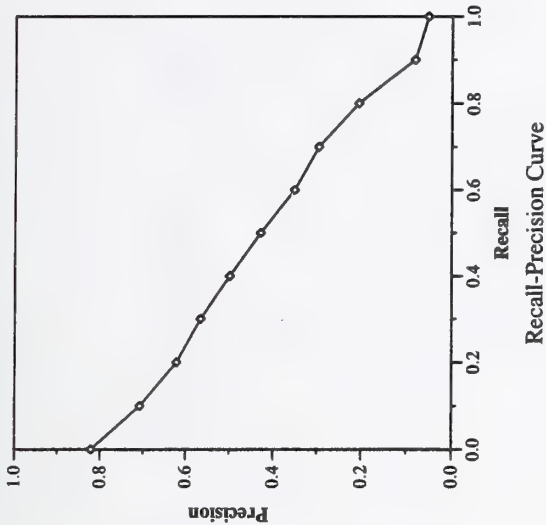
Document Level Averages	
	Precision
At 5 docs	0.6653
At 10 docs	0.5959
At 15 docs	0.5374
At 20 docs	0.4827
At 30 docs	0.3959
At 100 docs	0.1998
At 200 docs	0.1293
At 500 docs	0.0625
At 1000 docs	0.0327
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5066



Summary Statistics	
Run Number	cuhtk-cru-linsi-s1
Run Description	unknown, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1400

Recall Level Precision Averages	
Recall	Precision
0.00	0.8208
0.10	0.7082
0.20	0.6218
0.30	0.5655
0.40	0.4993
0.50	0.4285
0.60	0.3517
0.70	0.2969
0.80	0.2065
0.90	0.0815
1.00	0.0518
Average precision over all relevant docs	
non-interpolated	0.4019

Document Level Averages	
	Precision
At 5 docs	0.5959
At 10 docs	0.5204
At 15 docs	0.4435
At 20 docs	0.3898
At 30 docs	0.3272
At 100 docs	0.1731
At 200 docs	0.1115
At 500 docs	0.0533
At 1000 docs	0.0286
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4112

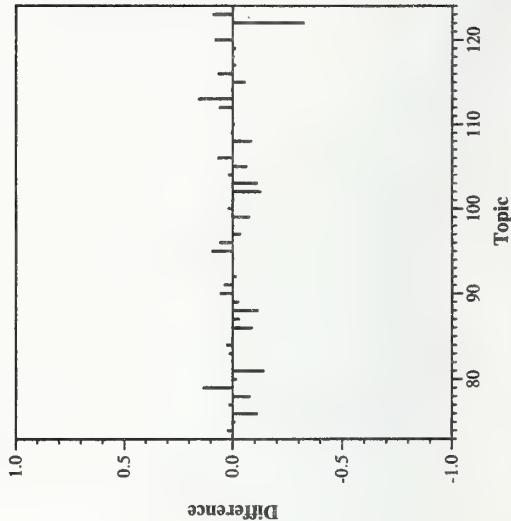
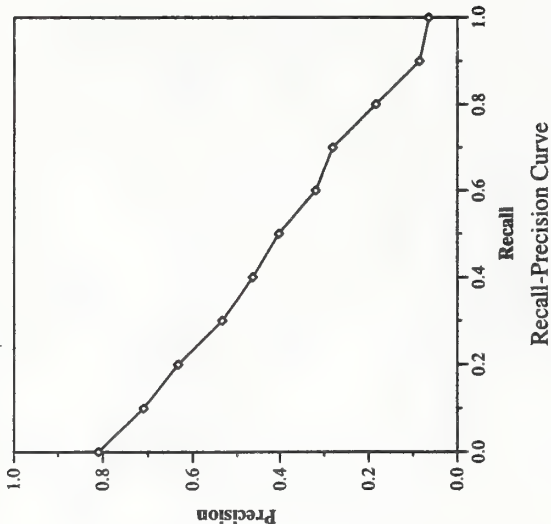


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	cuhtk-cru-nist-b2
Run Description	unknown, rolling
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1335

Recall Level Precision Averages	
Recall	Precision
0.00	0.8099
0.10	0.7103
0.20	0.6332
0.30	0.5342
0.40	0.4639
0.50	0.4030
0.60	0.3200
0.70	0.2812
0.80	0.1843
0.90	0.0856
1.00	0.0648
Average precision over all relevant docs	
non-interpolated	0.3870

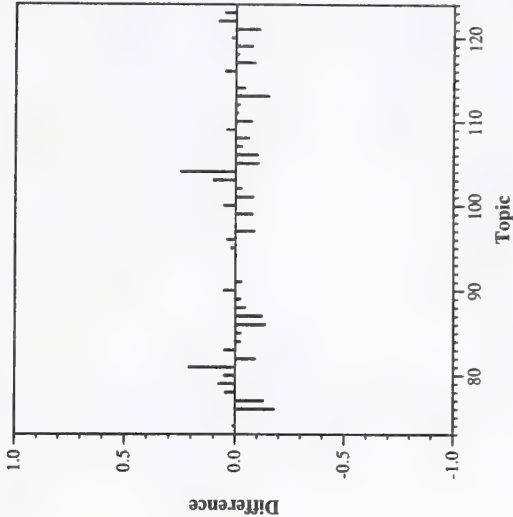
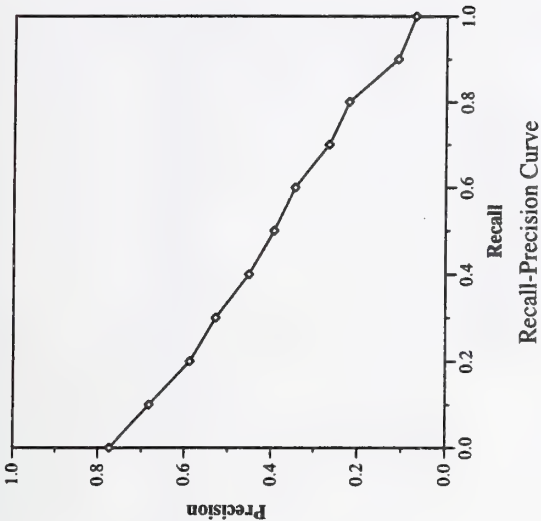
Document Level Averages	
At 5 docs	0.5837
At 10 docs	0.5245
At 15 docs	0.4476
At 20 docs	0.3918
At 30 docs	0.3204
At 100 docs	0.1678
At 200 docs	0.1068
At 500 docs	0.0507
At 1000 docs	0.0272
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4045



Summary Statistics	
Run Number	cuhtk-cru-shef-sl
Run Description	unknown, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1371

Recall Level Precision Averages	
Recall	Precision
0.00	0.7735
0.10	0.6824
0.20	0.5880
0.30	0.5286
0.40	0.4525
0.50	0.3946
0.60	0.3466
0.70	0.2669
0.80	0.2217
0.90	0.1108
1.00	0.0711
Average precision over all relevant docs	
non-interpolated	0.3824

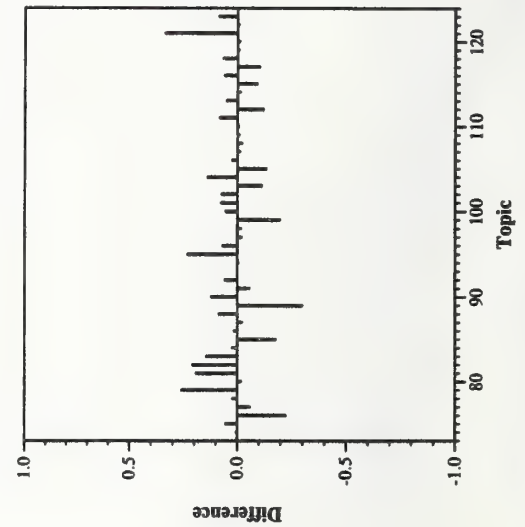
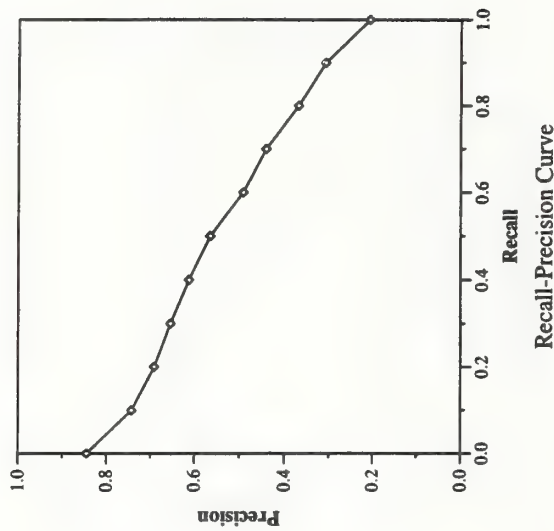
Document Level Averages	
	Precision
At 5 docs	0.5633
At 10 docs	0.4918
At 15 docs	0.4449
At 20 docs	0.3908
At 30 docs	0.3245
At 100 docs	0.1669
At 200 docs	0.1083
At 500 docs	0.0524
At 1000 docs	0.0280
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4110



Summary Statistics	
Run Number	cuhtk-r1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1637

Recall Level Precision Averages	
Recall	Precision
0.00	0.8446
0.10	0.7426
0.20	0.6914
0.30	0.6545
0.40	0.6129
0.50	0.5655
0.60	0.4926
0.70	0.4420
0.80	0.3702
0.90	0.3085
1.00	0.2077
Average precision over all relevant docs	
non-interpolated	0.5232

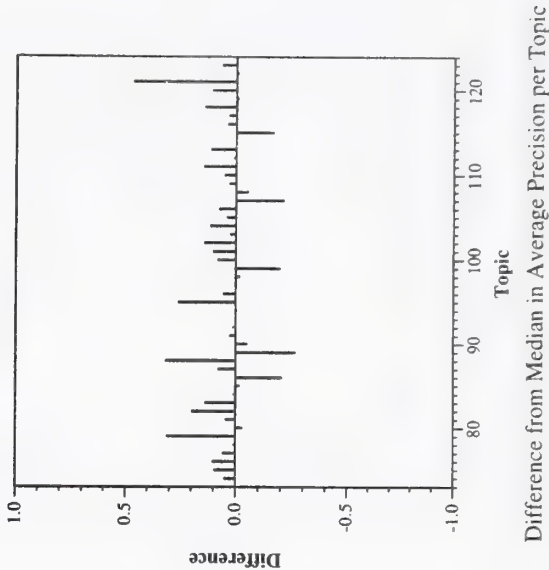
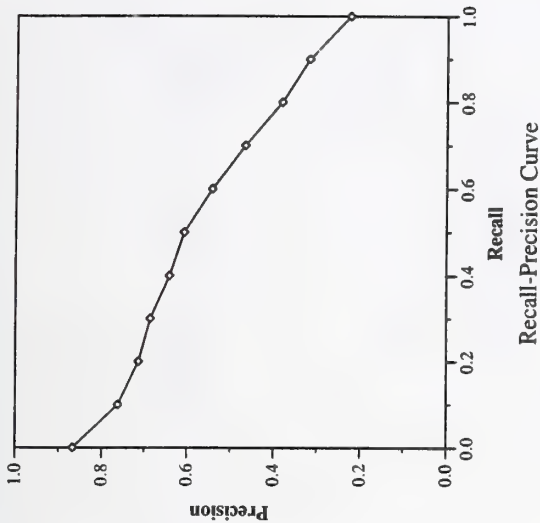
Document Level Averages	
	Precision
At 5 docs	0.6531
At 10 docs	0.5918
At 15 docs	0.5415
At 20 docs	0.4755
At 30 docs	0.3980
At 100 docs	0.2051
At 200 docs	0.1346
At 500 docs	0.0639
At 1000 docs	0.0334
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4974



Summary Statistics		
Run Number	cuhtk-s1	
Run Description	known, fixed	
Number of Topics	49	
Total number of documents over all topics		
Retrieved:	49000	
Relevant:	1818	
Rel-ret:	1642	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8670
0.10	0.7621
0.20	0.7140
0.30	0.6865
0.40	0.6417
0.50	0.6076
0.60	0.5433
0.70	0.4677
0.80	0.3827
0.90	0.3189
1.00	0.2252
Average precision over all relevant docs	
non-interpolated	0.5529

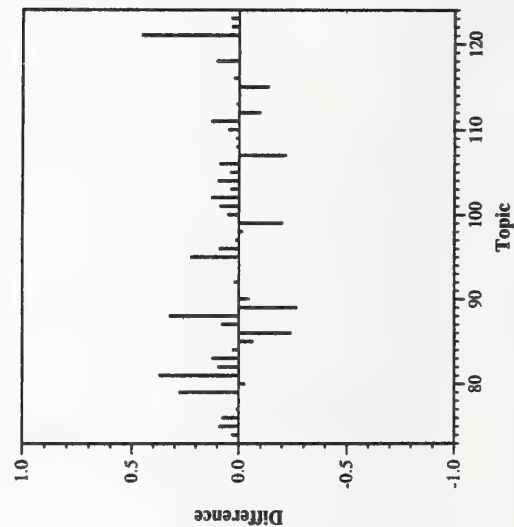
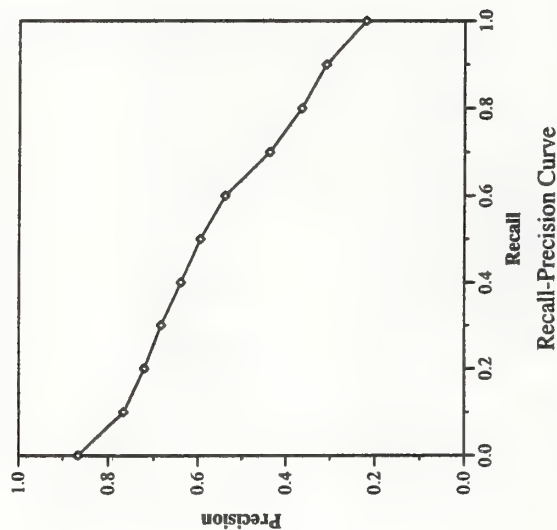
Document Level Averages	
At 5 docs	0.7347
At 10 docs	0.6163
At 15 docs	0.5592
At 20 docs	0.4898
At 30 docs	0.4034
At 100 docs	0.2080
At 200 docs	0.1353
At 500 docs	0.0639
At 1000 docs	0.0335
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5091



Summary Statistics		
Run Number	cuhk-slpl	
Run Description	known, fixed	
Number of Topics	49	
Total number of documents over all topics		
Retrieved:	49000	
Relevant:	1818	
Rel-ret:	1634	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8658
0.10	0.7663
0.20	0.7211
0.30	0.6839
0.40	0.6390
0.50	0.5950
0.60	0.5390
0.70	0.4375
0.80	0.3651
0.90	0.3098
1.00	0.2203
Average precision over all relevant docs	
non-interpolated	0.5451

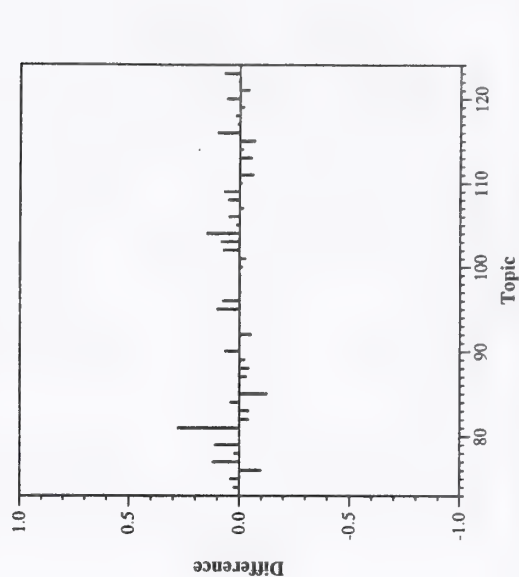
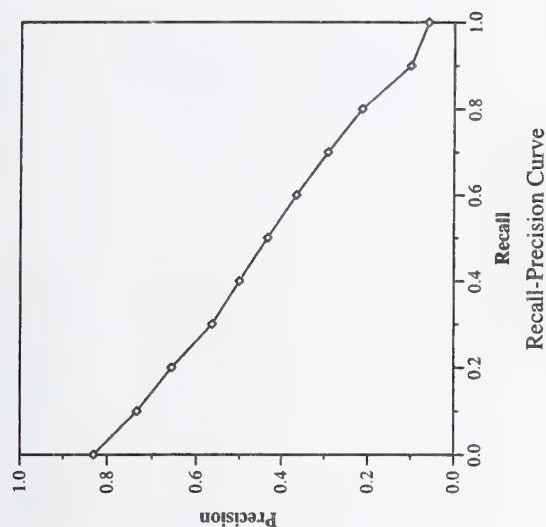
Document Level Averages	
	Precision
At 5 docs	0.7265
At 10 docs	0.6184
At 15 docs	0.5401
At 20 docs	0.4796
At 30 docs	0.3980
At 100 docs	0.2027
At 200 docs	0.1344
At 500 docs	0.0634
At 1000 docs	0.0333
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5219



Summary Statistics	
Run Number	cuhtk-s1plu
Run Description	unknown, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1411

Recall Level Precision Averages	
Recall	Precision
0.00	0.8306
0.10	0.7341
0.20	0.6560
0.30	0.5643
0.40	0.5008
0.50	0.4334
0.60	0.3655
0.70	0.2922
0.80	0.2138
0.90	0.1009
1.00	0.0598
Average precision over all relevant docs	
non-interpolated	0.4150

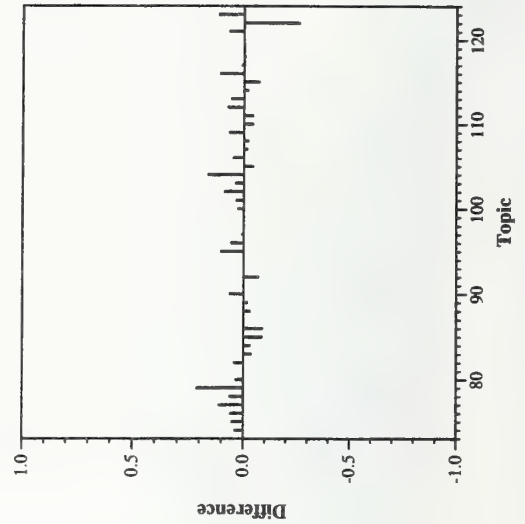
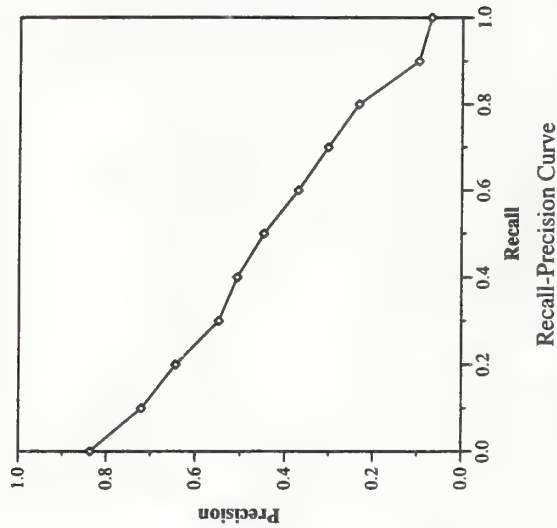
Document Level Averages	
	Precision
At 5 docs	0.6122
At 10 docs	0.5306
At 15 docs	0.4517
At 20 docs	0.4082
At 30 docs	0.3286
At 100 docs	0.1747
At 200 docs	0.1118
At 500 docs	0.0547
At 1000 docs	0.0288
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4163



Summary Statistics	
Run Number	cuhk-slu
Run Description	unknown, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1426

Recall Level Precision Averages	
Recall	Precision
0.00	0.8371
0.10	0.7206
0.20	0.6440
0.30	0.5489
0.40	0.5082
0.50	0.4485
0.60	0.3715
0.70	0.3035
0.80	0.2341
0.90	0.0983
1.00	0.0699
Average precision over all relevant docs	
non-interpolated	0.4147

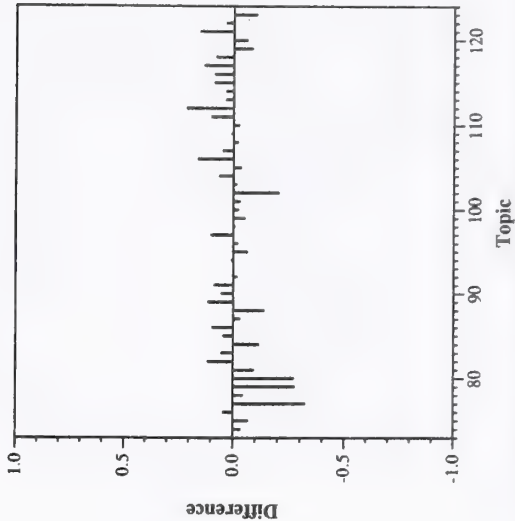
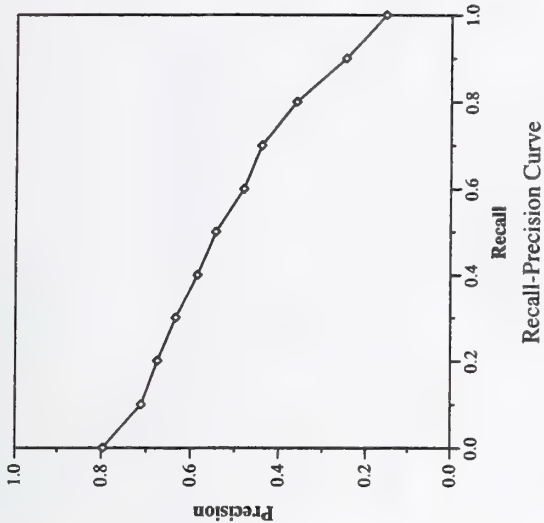
Document Level Averages	
	Precision
At 5 docs	0.5878
At 10 docs	0.5286
At 15 docs	0.4612
At 20 docs	0.4041
At 30 docs	0.3395
At 100 docs	0.1761
At 200 docs	0.1135
At 500 docs	0.0553
At 1000 docs	0.0291
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4198



Summary Statistics		
Run Number	cmu-b1	
Run Description	known, rolling	
Number of Topics	49	
Total number of documents over all topics		
Retrieved:	49000	
Relevant:	1818	
Rel-ret:	1626	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7964
0.10	0.7118
0.20	0.6757
0.30	0.6348
0.40	0.5855
0.50	0.5441
0.60	0.4813
0.70	0.4412
0.80	0.3624
0.90	0.2485
1.00	0.1546
Average precision over all relevant docs	
non-interpolated	0.5018

Document Level Averages	
	Precision
At 5 docs	0.6367
At 10 docs	0.5816
At 15 docs	0.5170
At 20 docs	0.4735
At 30 docs	0.3959
At 100 docs	0.2092
At 200 docs	0.1308
At 500 docs	0.0602
At 1000 docs	0.0332
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4998

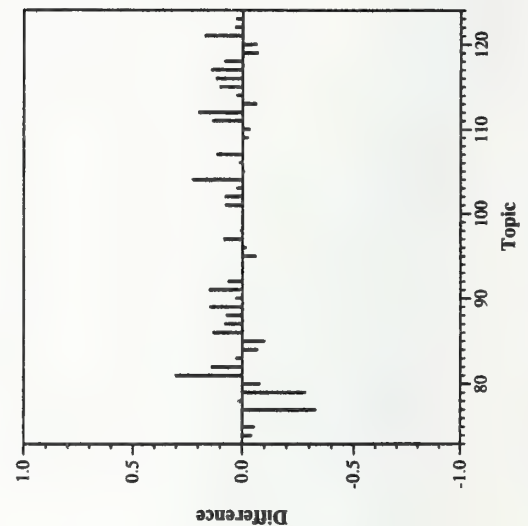
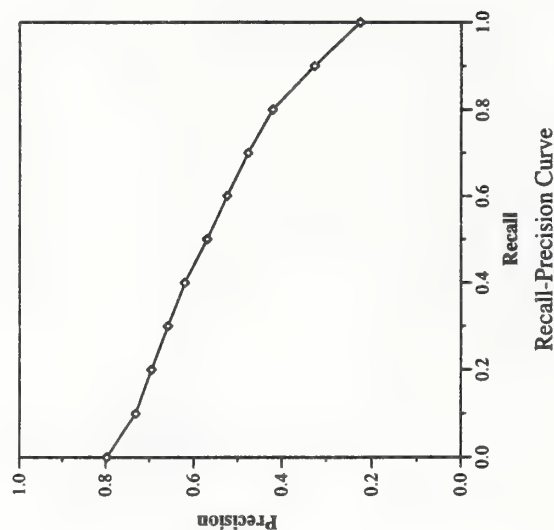


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	cmu-r1
Run Description	known, rolling
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1689

Recall Level Precision Averages	
Recall	Precision
0.00	0.7976
0.10	0.7311
0.20	0.6951
0.30	0.6581
0.40	0.6204
0.50	0.5698
0.60	0.5245
0.70	0.4768
0.80	0.4215
0.90	0.3263
1.00	0.2255
Average precision over all relevant docs	
non-interpolated	0.5377

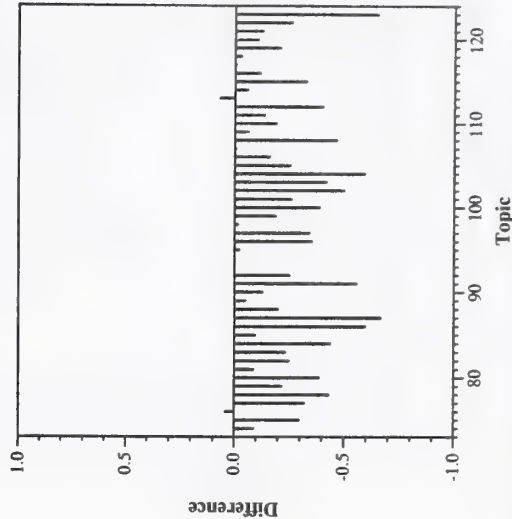
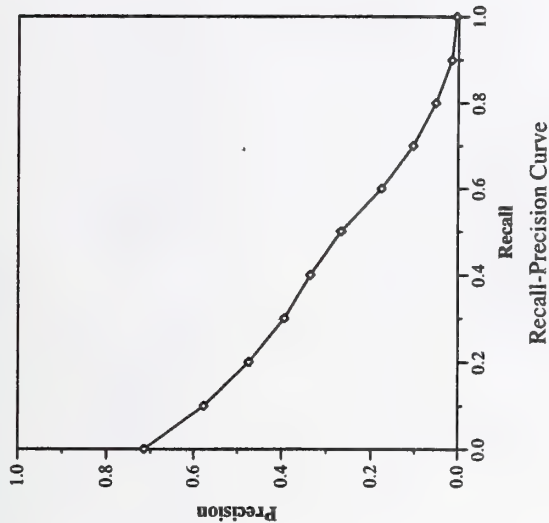
Document Level Averages	
	Precision
At 5 docs	0.6449
At 10 docs	0.6082
At 15 docs	0.5401
At 20 docs	0.4949
At 30 docs	0.4163
At 100 docs	0.2167
At 200 docs	0.1400
At 500 docs	0.0644
At 1000 docs	0.0345
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5349



Summary Statistics	
Run Number	cmu-s1
Run Description	known, rolling
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1193

Recall Level Precision Averages	
Recall	Precision
0.00	0.7144
0.10	0.5769
0.20	0.4745
0.30	0.3939
0.40	0.3354
0.50	0.2675
0.60	0.1769
0.70	0.1044
0.80	0.0521
0.90	0.0152
1.00	0.0043
Average precision over all relevant docs	
non-interpolated	0.2653

Document Level Averages	
	Precision
At 5 docs	0.4735
At 10 docs	0.4000
At 15 docs	0.3388
At 20 docs	0.3010
At 30 docs	0.2517
At 100 docs	0.1316
At 200 docs	0.0840
At 500 docs	0.0422
At 1000 docs	0.0243
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2907

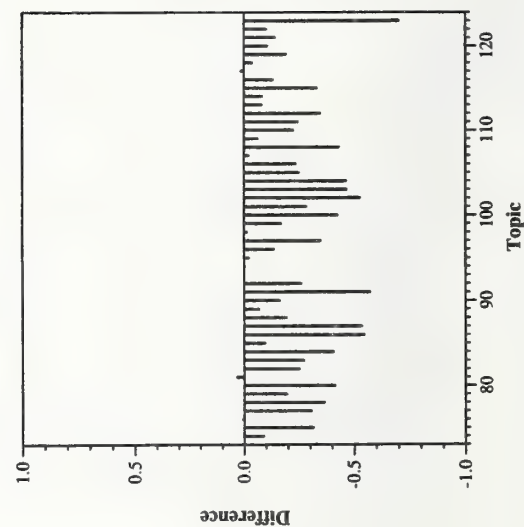
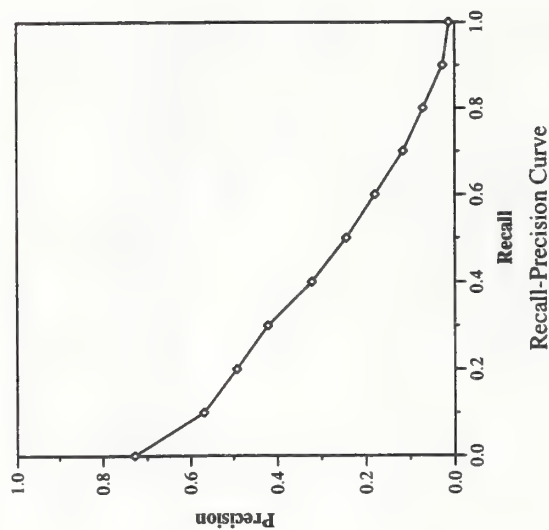


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	cmu-s2
Run Description	known, rolling
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1165

Recall Level Precision Averages	
Recall	Precision
0.00	0.7288
0.10	0.5693
0.20	0.4919
0.30	0.4211
0.40	0.3211
0.50	0.2436
0.60	0.1792
0.70	0.1151
0.80	0.0703
0.90	0.0267
1.00	0.0130
Average precision over all relevant docs	
non-interpolated	0.2706

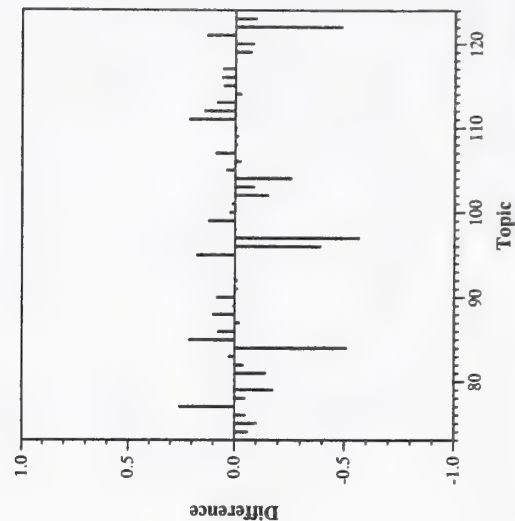
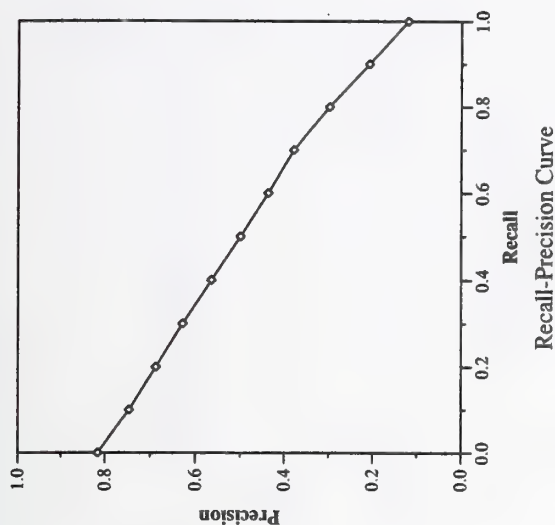
Document Level Averages	
	Precision
At 5 docs	0.4694
At 10 docs	0.3918
At 15 docs	0.3306
At 20 docs	0.2939
At 30 docs	0.2490
At 100 docs	0.1302
At 200 docs	0.0822
At 500 docs	0.0409
At 1000 docs	0.0238
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3021



Summary Statistics		
Run Number	ibms-b1	
Run Description	known, fixed	
Number of Topics	49	
Total number of documents over all topics		
Retrieved:	49000	
Relevant:	1818	
Rel-ret:	1625	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8161
0.10	0.7465
0.20	0.6877
0.30	0.6286
0.40	0.5646
0.50	0.4998
0.60	0.4373
0.70	0.3790
0.80	0.2984
0.90	0.2081
1.00	0.1199
Average precision over all relevant docs	
non-interpolated	0.4765

Document Level Averages	
	Precision
At 5 docs	0.6694
At 10 docs	0.5776
At 15 docs	0.5007
At 20 docs	0.4561
At 30 docs	0.3857
At 100 docs	0.2080
At 200 docs	0.1302
At 500 docs	0.0598
At 1000 docs	0.0332
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4732

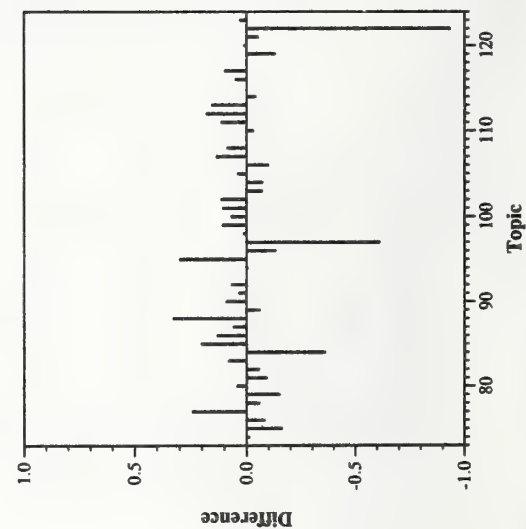
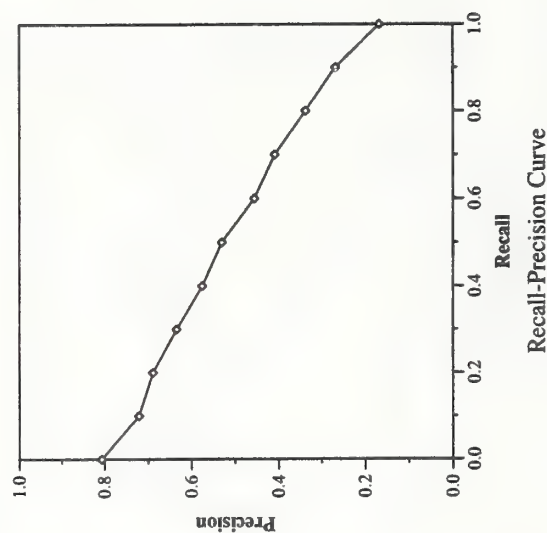


Spoken document retrieval track results — IBM T.J. Watson Research Center (Franz)

Summary Statistics	
Run Number	ibms-r1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1684

Recall Level Precision Averages	
Recall	Precision
0.00	0.8063
0.10	0.7212
0.20	0.6892
0.30	0.6344
0.40	0.5753
0.50	0.5304
0.60	0.4563
0.70	0.4092
0.80	0.3379
0.90	0.2681
1.00	0.1681
Average precision over all relevant docs	
non-interpolated	0.4994

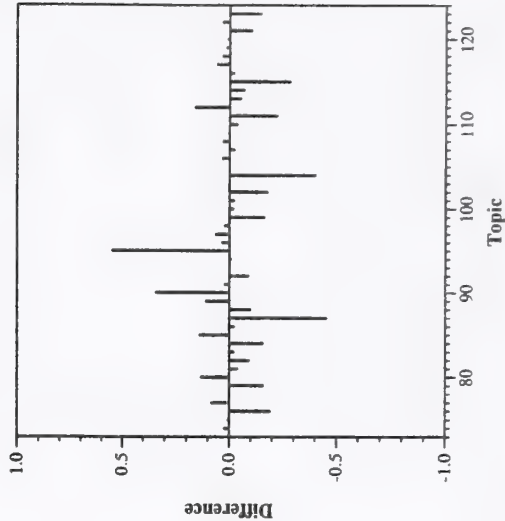
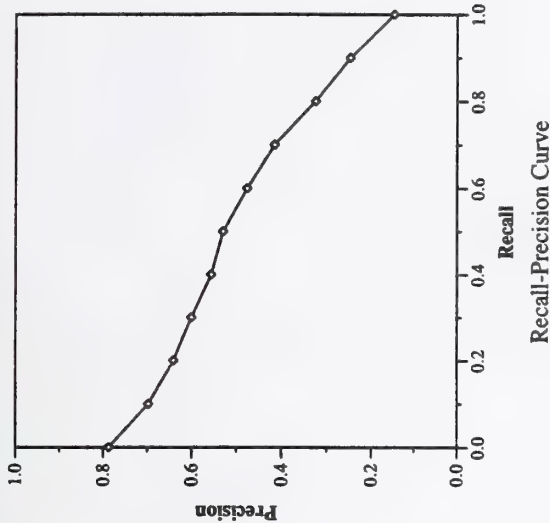
Document Level Averages	
	Precision
At 5 docs	0.6367
At 10 docs	0.5612
At 15 docs	0.5020
At 20 docs	0.4622
At 30 docs	0.3952
At 100 docs	0.2108
At 200 docs	0.1382
At 500 docs	0.0635
At 1000 docs	0.0344
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4791



Summary Statistics	
Run Number	limsi-b1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1586

Recall Level Precision Averages	
Recall	Precision
0.00	0.7877
0.10	0.6974
0.20	0.6409
0.30	0.6006
0.40	0.5560
0.50	0.5293
0.60	0.4761
0.70	0.4154
0.80	0.3252
0.90	0.2480
1.00	0.1465
Average precision over all relevant docs	
non-interpolated	0.4828

Document Level Averages	
	Precision
At 5 docs	0.6286
At 10 docs	0.5571
At 15 docs	0.5156
At 20 docs	0.4622
At 30 docs	0.3830
At 100 docs	0.2041
At 200 docs	0.1278
At 500 docs	0.0593
At 1000 docs	0.0324
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4704

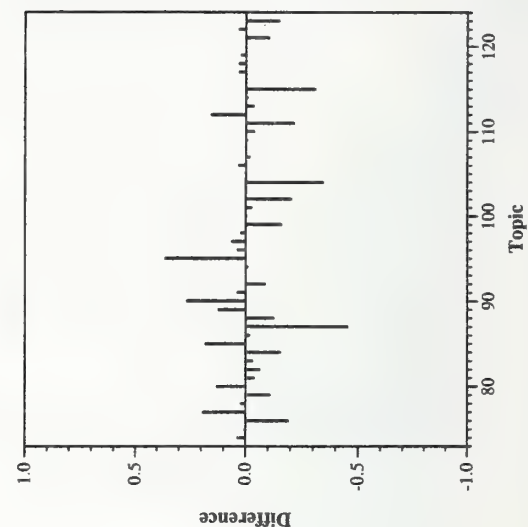
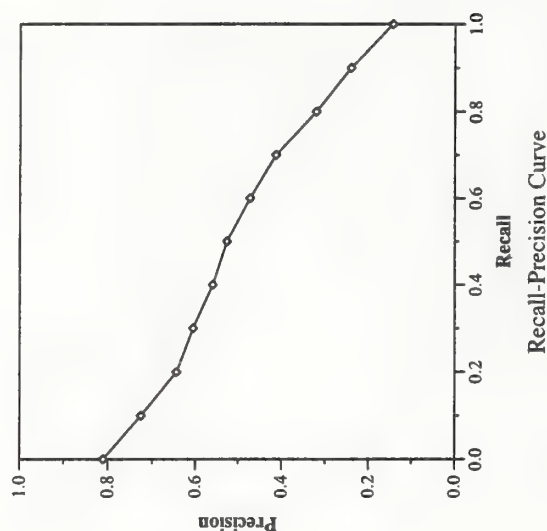


Spoken document retrieval track results — LIMSI-CNRS

Summary Statistics	
Run Number	limsi-b2
Run Description	known, rolling
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1568

Recall Level Precision Averages	
Recall	Precision
0.00	0.8101
0.10	0.7260
0.20	0.6445
0.30	0.6058
0.40	0.5597
0.50	0.5265
0.60	0.4730
0.70	0.4137
0.80	0.3215
0.90	0.2413
1.00	0.1438
Average precision over all relevant docs	
non-interpolated	0.4839

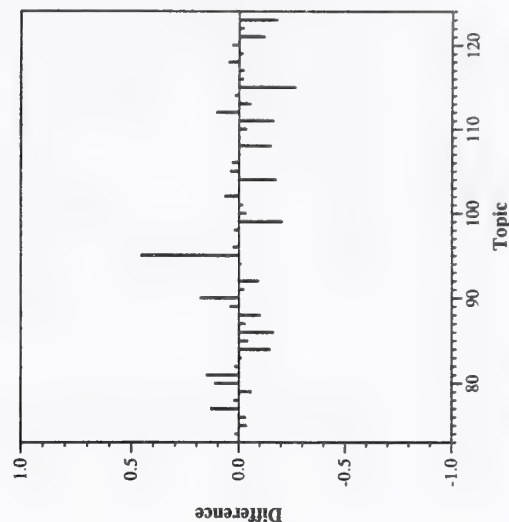
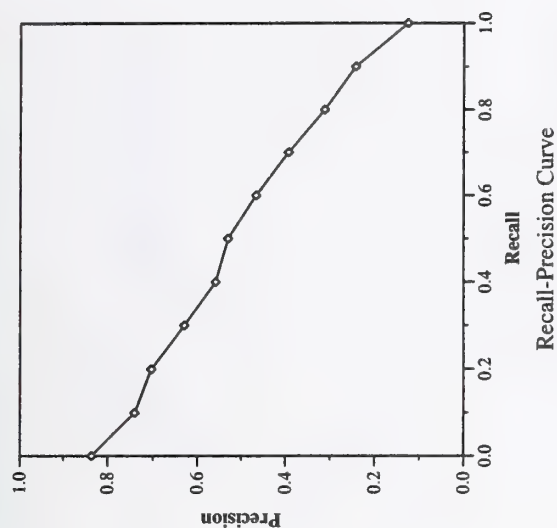
Document Level Averages	
At 5 docs	0.6327
At 10 docs	0.5694
At 15 docs	0.5170
At 20 docs	0.4684
At 30 docs	0.3830
At 100 docs	0.2049
At 200 docs	0.1271
At 500 docs	0.0588
At 1000 docs	0.0320
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4776



Summary Statistics	
Run Number	limsi-cr-att1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1616

Recall Level Precision Averages	
Recall	Precision
0.00	0.8355
0.10	0.7406
0.20	0.7036
0.30	0.6297
0.40	0.5584
0.50	0.5304
0.60	0.4667
0.70	0.3934
0.80	0.3122
0.90	0.2430
1.00	0.1255
Average precision over all relevant docs	
non-interpolated	0.4925

Document Level Averages	
	Precision
At 5 docs	0.6490
At 10 docs	0.5714
At 15 docs	0.4952
At 20 docs	0.4469
At 30 docs	0.3660
At 100 docs	0.1996
At 200 docs	0.1313
At 500 docs	0.0615
At 1000 docs	0.0330
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4851

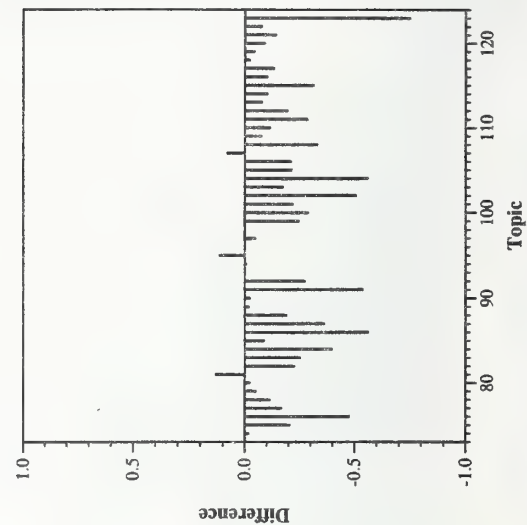
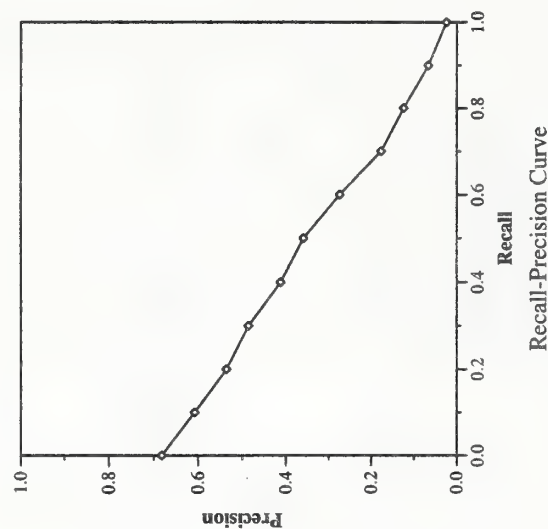


Spoken document retrieval track results — LIMSI-CNRS

Summary Statistics	
Run Number	limsi-cr-cmul
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1259

Recall Level Precision Averages	
Recall	Precision
0.00	0.6818
0.10	0.6079
0.20	0.5365
0.30	0.4862
0.40	0.4122
0.50	0.3592
0.60	0.2738
0.70	0.1766
0.80	0.1239
0.90	0.0664
1.00	0.0244
Average precision over all relevant docs	
non-interpolated	0.3234

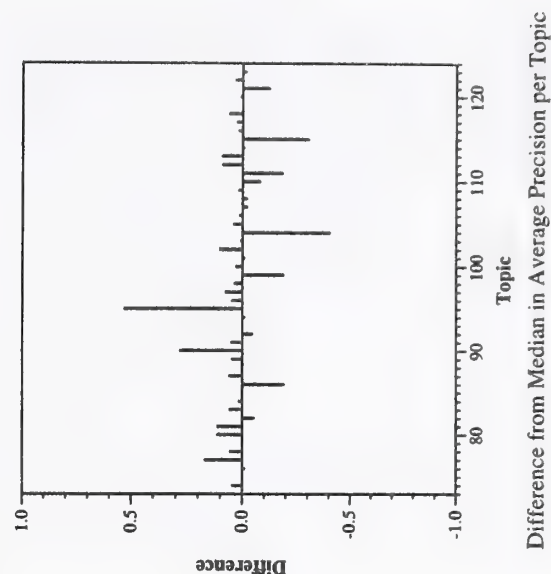
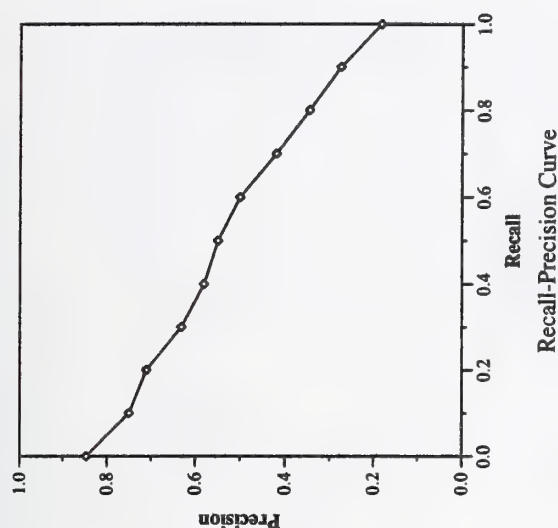
Document Level Averages	
At 5 docs	0.4939
At 10 docs	0.4082
At 15 docs	0.3769
At 20 docs	0.3459
At 30 docs	0.2871
At 100 docs	0.1508
At 200 docs	0.0943
At 500 docs	0.0460
At 1000 docs	0.0257
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3551



Summary Statistics	
Run Number	limsi-cr-cultkl
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1644

Recall Level Precision Averages	
Recall	Precision
0.00	0.8468
0.10	0.7505
0.20	0.7107
0.30	0.6312
0.40	0.5804
0.50	0.5494
0.60	0.5006
0.70	0.4198
0.80	0.3466
0.90	0.2757
1.00	0.1836
Average precision over all relevant docs	
non-interpolated	0.5176

Document Level Averages	
	Precision
At 5 docs	0.6490
At 10 docs	0.5796
At 15 docs	0.5211
At 20 docs	0.4714
At 30 docs	0.3932
At 100 docs	0.2063
At 200 docs	0.1350
At 500 docs	0.0631
At 1000 docs	0.0336
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5014

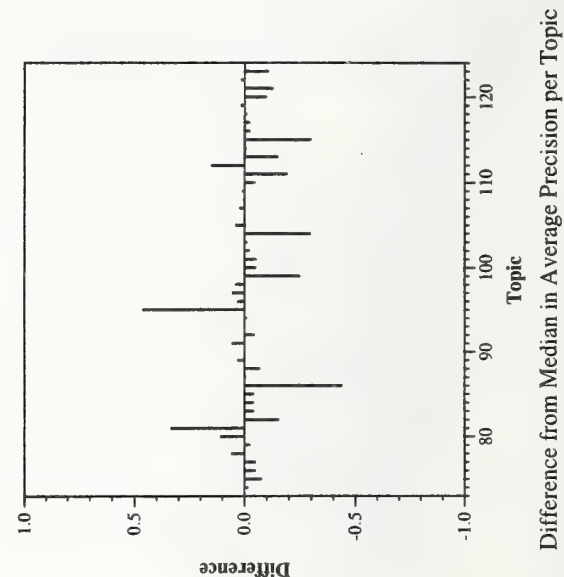
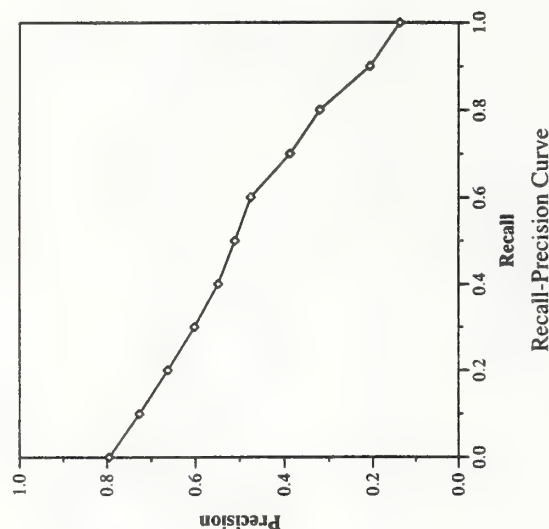


Spoken document retrieval track results — LIMSI-CNRS

Summary Statistics	
Run Number	limsi-cr-shef1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1562

Recall Level Precision Averages	
Recall	Precision
0.00	0.7951
0.10	0.7270
0.20	0.6627
0.30	0.6019
0.40	0.5489
0.50	0.5110
0.60	0.4755
0.70	0.3877
0.80	0.3211
0.90	0.2071
1.00	0.1374
Average precision over all relevant docs	
non-interpolated	0.4787

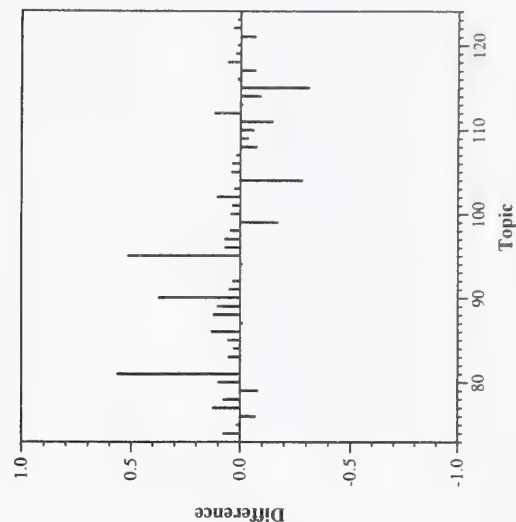
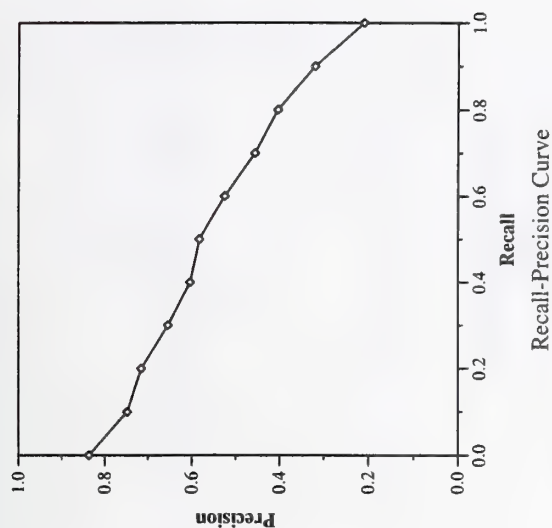
Document Level Averages	
	Precision
At 5 docs	0.6041
At 10 docs	0.5510
At 15 docs	0.4980
At 20 docs	0.4480
At 30 docs	0.3667
At 100 docs	0.1947
At 200 docs	0.1260
At 500 docs	0.0592
At 1000 docs	0.0319
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4706



Summary Statistics	
Run Number	limsi-r1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1662

Recall Level Precision Averages	
Recall	Precision
0.00	0.8368
0.10	0.7483
0.20	0.7152
0.30	0.6537
0.40	0.6033
0.50	0.5828
0.60	0.5252
0.70	0.4562
0.80	0.4043
0.90	0.3206
1.00	0.2104
Average precision over all relevant docs	
non-interpolated	0.5411

Document Level Averages	
	Precision
At 5 docs	0.6531
At 10 docs	0.6143
At 15 docs	0.5442
At 20 docs	0.4929
At 30 docs	0.3980
At 100 docs	0.2100
At 200 docs	0.1369
At 500 docs	0.0634
At 1000 docs	0.0339
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5239

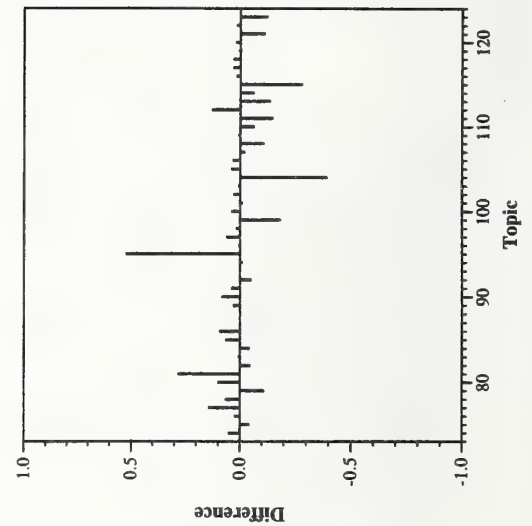
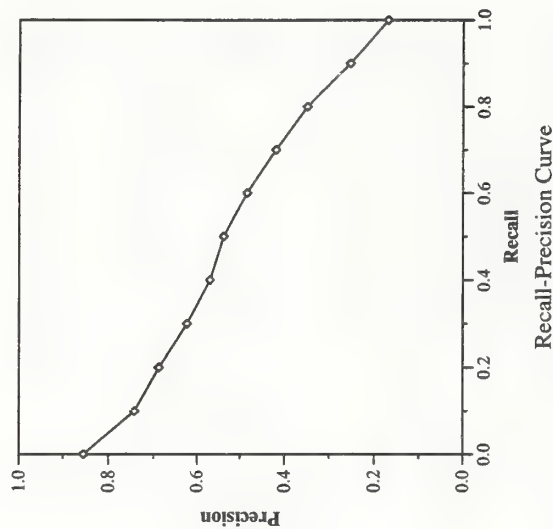


Spoken document retrieval track results — LIMSI-CNRS

Summary Statistics	
Run Number	limsi-sl
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1612

Recall Level Precision Averages	
Recall	Precision
0.00	0.8549
0.10	0.7427
0.20	0.6883
0.30	0.6238
0.40	0.5700
0.50	0.5380
0.60	0.4852
0.70	0.4200
0.80	0.3497
0.90	0.2542
1.00	0.1701
Average precision over all relevant docs	
non-interpolated	0.5072

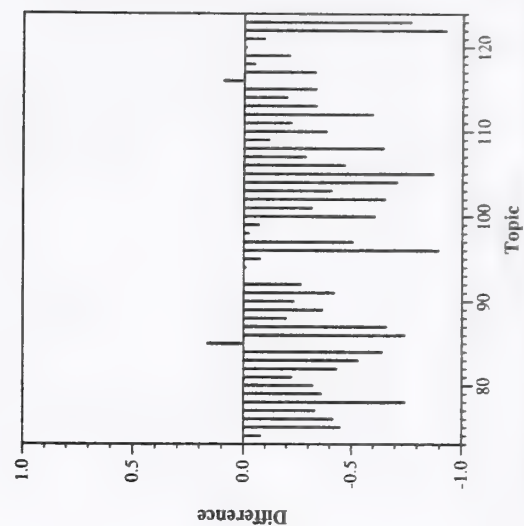
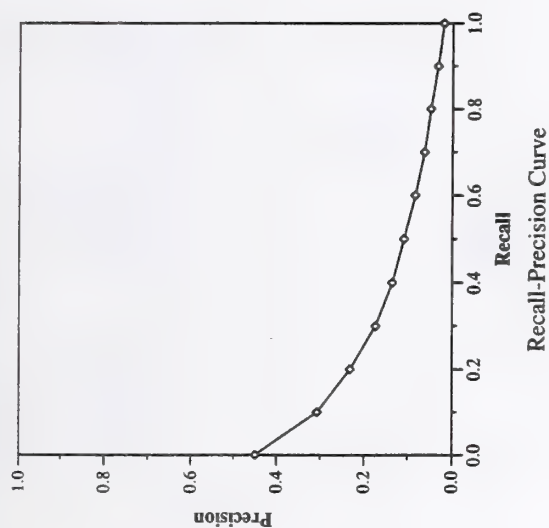
Document Level Averages	
	Precision
At 5 docs	0.6408
At 10 docs	0.5816
At 15 docs	0.5184
At 20 docs	0.4653
At 30 docs	0.3762
At 100 docs	0.2031
At 200 docs	0.1320
At 500 docs	0.0613
At 1000 docs	0.0329
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4876



Summary Statistics	
Run Number	mds08-b1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1251

Recall Level Precision Averages	
Recall	Precision
0.00	0.4493
0.10	0.3076
0.20	0.2324
0.30	0.1741
0.40	0.1352
0.50	0.1087
0.60	0.0833
0.70	0.0629
0.80	0.0487
0.90	0.0325
1.00	0.0199
Average precision over all relevant docs	
non-interpolated	0.1350

Document Level Averages	
	Precision
At 5 docs	0.2367
At 10 docs	0.2061
At 15 docs	0.1782
At 20 docs	0.1653
At 30 docs	0.1320
At 100 docs	0.0827
At 200 docs	0.0619
At 500 docs	0.0374
At 1000 docs	0.0255
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1522

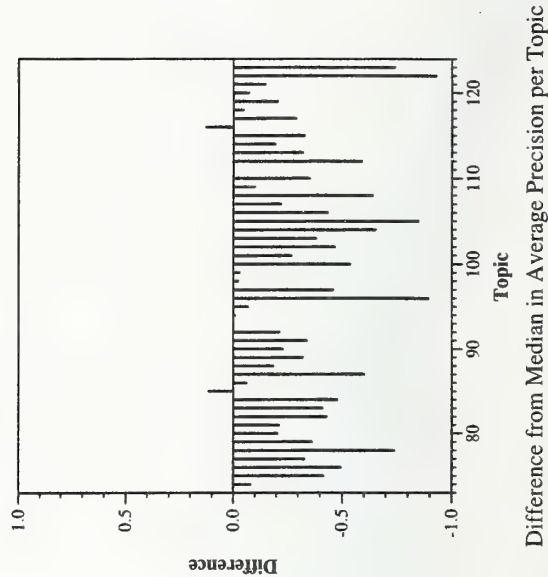
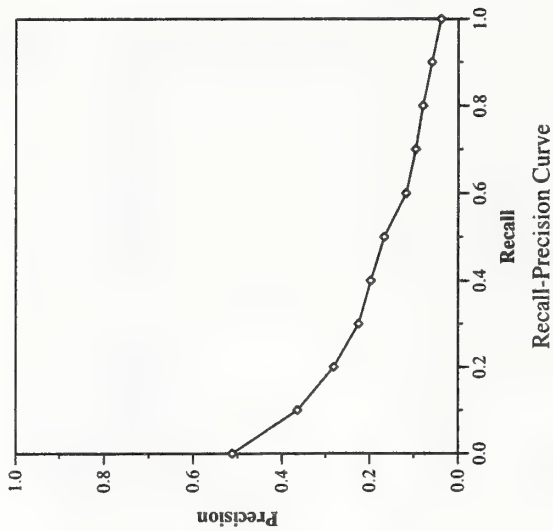


Spoken document retrieval track results — RMIT

Summary Statistics	
Run Number	mds08-r1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1398

Recall Level Precision Averages	
Recall	Precision
0.00	0.5125
0.10	0.3647
0.20	0.2815
0.30	0.2250
0.40	0.1974
0.50	0.1671
0.60	0.1174
0.70	0.0958
0.80	0.0799
0.90	0.0592
1.00	0.0384
Average precision over all relevant docs	
non-interpolated	0.1775

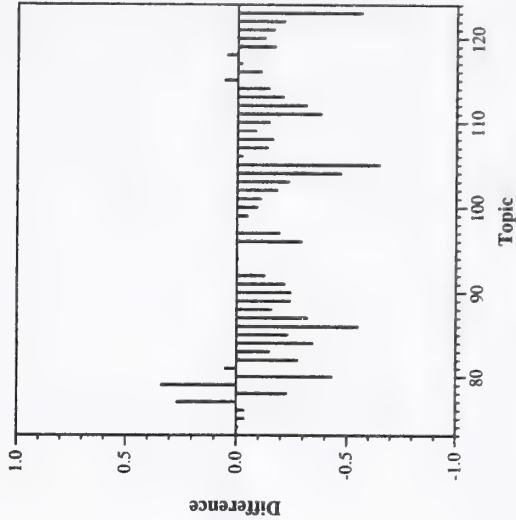
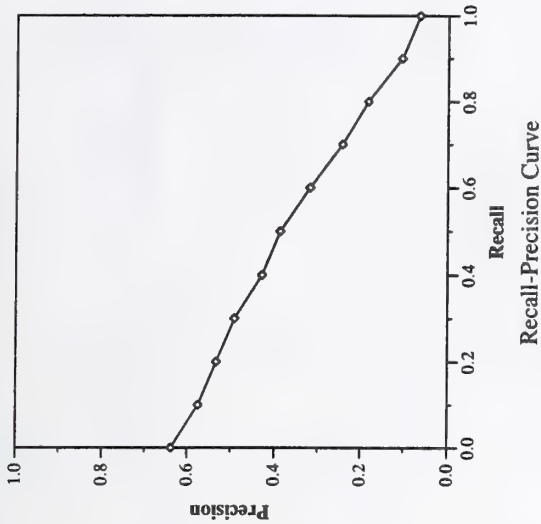
Document Level Averages	
	Precision
At 5 docs	0.2980
At 10 docs	0.2633
At 15 docs	0.2177
At 20 docs	0.2000
At 30 docs	0.1680
At 100 docs	0.0978
At 200 docs	0.0688
At 500 docs	0.0425
At 1000 docs	0.0285
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1891



Summary Statistics	
Run Number	cedar-b1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1468

Recall Level Precision Averages	
Recall	Precision
0.00	0.6378
0.10	0.5745
0.20	0.5329
0.30	0.4913
0.40	0.4289
0.50	0.3876
0.60	0.3195
0.70	0.2442
0.80	0.1833
0.90	0.1055
1.00	0.0650
Average precision over all relevant docs	
non-interpolated	0.3430

Document Level Averages	
	Precision
At 5 docs	0.4245
At 10 docs	0.4000
At 15 docs	0.3633
At 20 docs	0.3286
At 30 docs	0.2878
At 100 docs	0.1733
At 200 docs	0.1127
At 500 docs	0.0551
At 1000 docs	0.0300
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3572

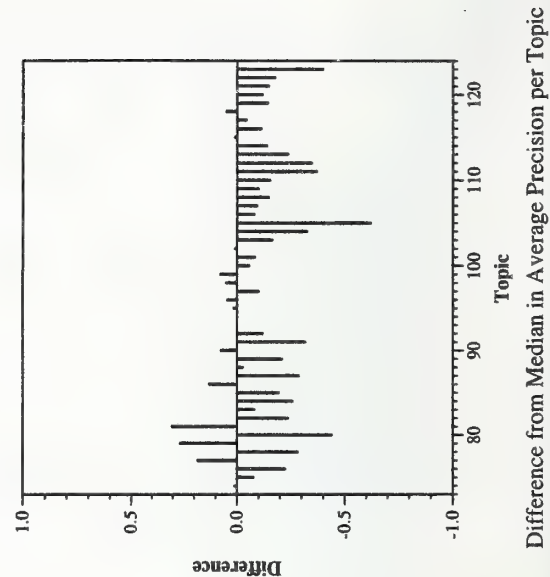
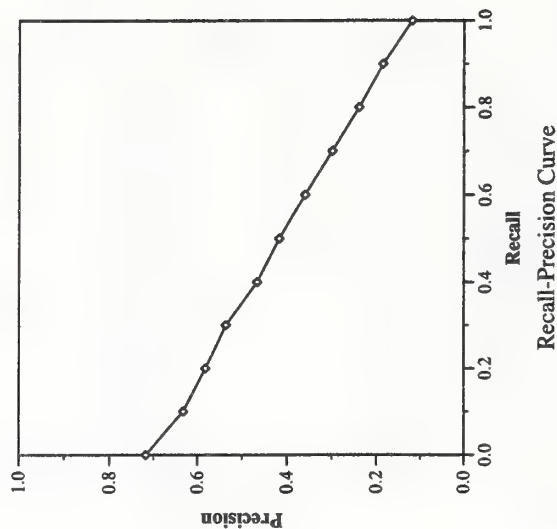


Spoken document retrieval track results — State University of New York at Buffalo

Summary Statistics	
Run Number	cedar-r1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1586

Recall Level Precision Averages	
Recall	Precision
0.00	0.7174
0.10	0.6326
0.20	0.5819
0.30	0.5358
0.40	0.4667
0.50	0.4178
0.60	0.3605
0.70	0.2998
0.80	0.2396
0.90	0.1854
1.00	0.1192
Average precision over all relevant docs	
non-interpolated	0.3906

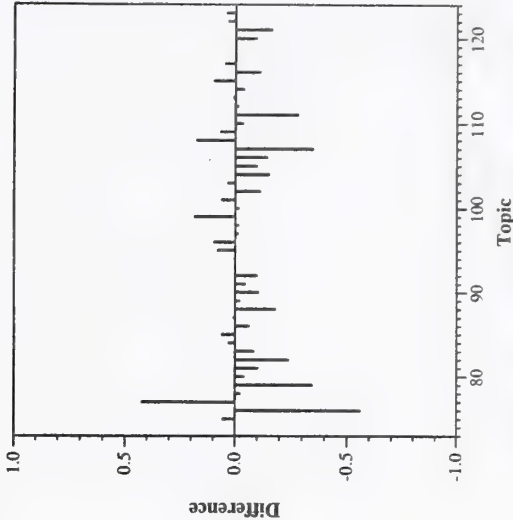
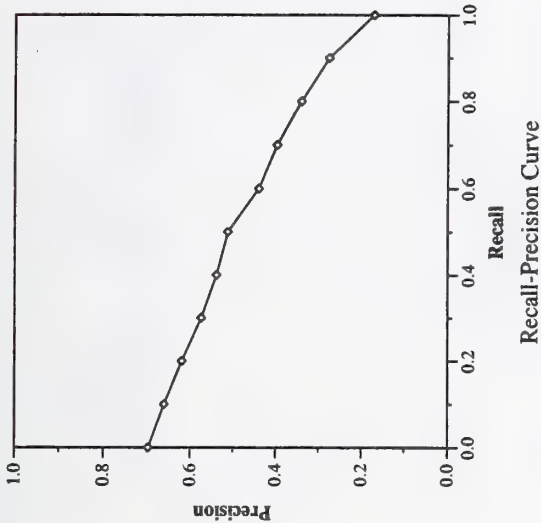
Document Level Averages	
	Precision
At 5 docs	0.4816
At 10 docs	0.4551
At 15 docs	0.4082
At 20 docs	0.3612
At 30 docs	0.3109
At 100 docs	0.1841
At 200 docs	0.1247
At 500 docs	0.0600
At 1000 docs	0.0324
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3961



Summary Statistics	
Run Number	tno8b-b1-limsi
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1612

Recall Level Precision Averages	
Recall	Precision
0.00	0.6973
0.10	0.6604
0.20	0.6193
0.30	0.5742
0.40	0.5387
0.50	0.5121
0.60	0.4398
0.70	0.3971
0.80	0.3407
0.90	0.2764
1.00	0.1719
Average precision over all relevant docs	
non-interpolated	0.4650

Document Level Averages	
At 5 docs	0.5265
At 10 docs	0.5204
At 15 docs	0.4830
At 20 docs	0.4398
At 30 docs	0.3565
At 100 docs	0.2000
At 200 docs	0.1289
At 500 docs	0.0595
At 1000 docs	0.0329
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4469

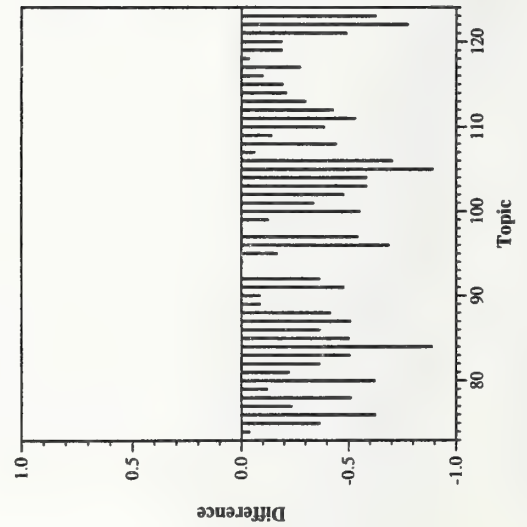
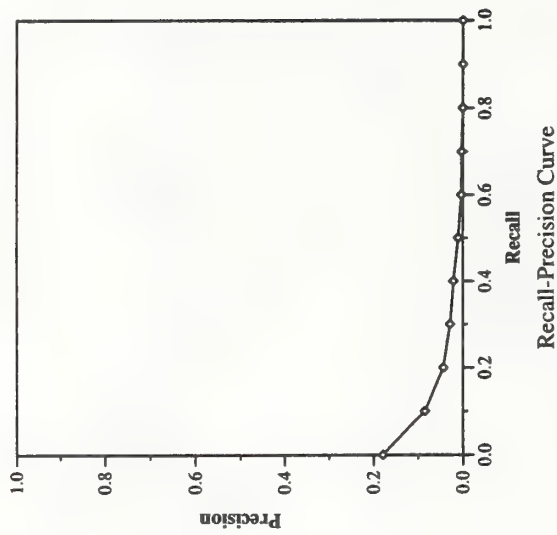


Spoken document retrieval track results — TwentyOne

Summary Statistics	
Run Number	tno8b-bl1u-limsi
Run Description	unknown, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	480

Recall Level Precision Averages	
Recall	Precision
0.00	0.1793
0.10	0.0847
0.20	0.0437
0.30	0.0287
0.40	0.0210
0.50	0.0105
0.60	0.0036
0.70	0.0030
0.80	0.0005
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0238

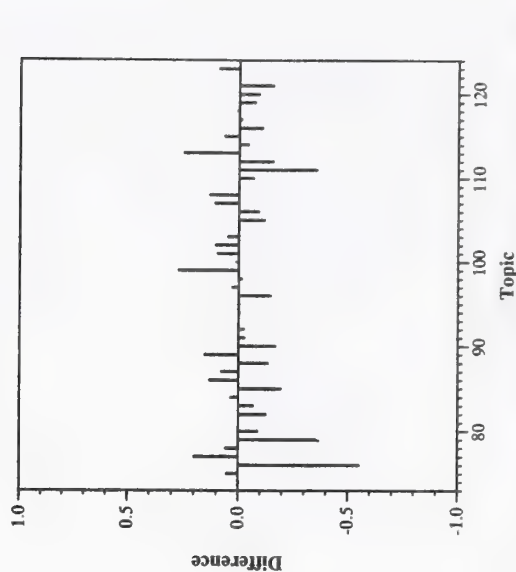
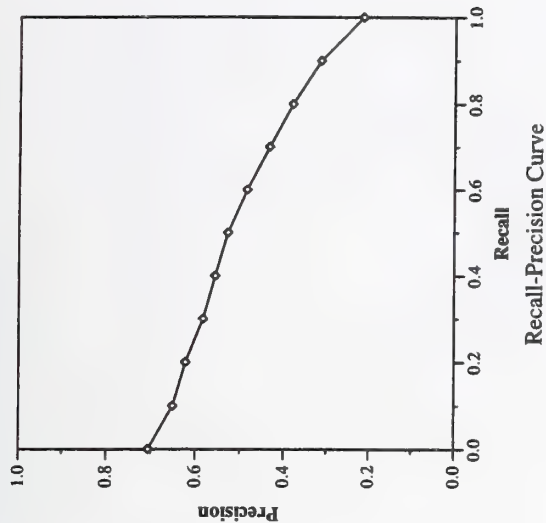
Document Level Averages	
	Precision
At 5 docs	0.0612
At 10 docs	0.0510
At 15 docs	0.0408
At 20 docs	0.0367
At 30 docs	0.0340
At 100 docs	0.0273
At 200 docs	0.0213
At 500 docs	0.0152
At 1000 docs	0.0098
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0311



Summary Statistics	
Run Number	tno8b-r1-limsi
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1645

Recall Level Precision Averages	
Recall	Precision
0.00	0.7061
0.10	0.6507
0.20	0.6225
0.30	0.5831
0.40	0.5564
0.50	0.5283
0.60	0.4855
0.70	0.4351
0.80	0.3820
0.90	0.3166
1.00	0.2162
Average precision over all relevant docs	
non-interpolated	0.4806

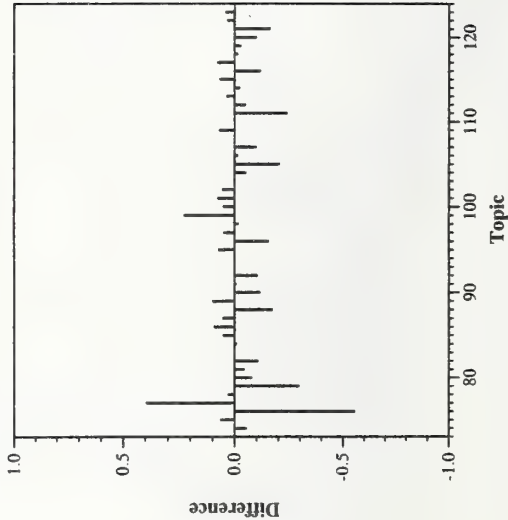
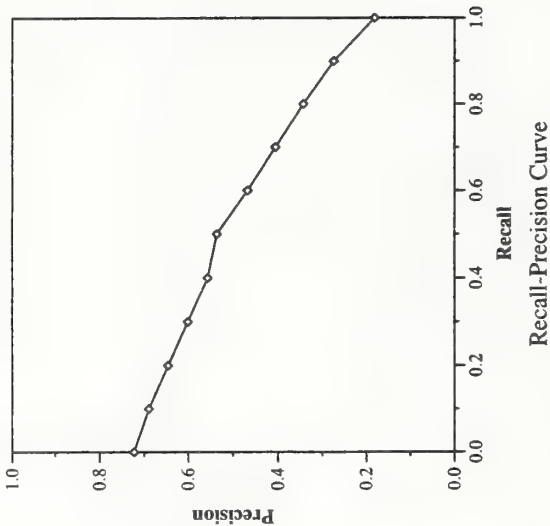
Document Level Averages	
	Precision
At 5 docs	0.5265
At 10 docs	0.5265
At 15 docs	0.4857
At 20 docs	0.4418
At 30 docs	0.3667
At 100 docs	0.1996
At 200 docs	0.1351
At 500 docs	0.0624
At 1000 docs	0.0336
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4613



Summary Statistics	
Run Number	tno8b-s1-limsi
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1631

Recall Level Precision Averages	
Recall	Precision
0.00	0.7227
0.10	0.6891
0.20	0.6455
0.30	0.6008
0.40	0.5560
0.50	0.5356
0.60	0.4675
0.70	0.4056
0.80	0.3436
0.90	0.2750
1.00	0.1814
Average precision over all relevant docs	
non-interpolated	0.4826

Document Level Averages	
	Precision
At 5 docs	0.5796
At 10 docs	0.5408
At 15 docs	0.4884
At 20 docs	0.4531
At 30 docs	0.3694
At 100 docs	0.2016
At 200 docs	0.1301
At 500 docs	0.0606
At 1000 docs	0.0333
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4647

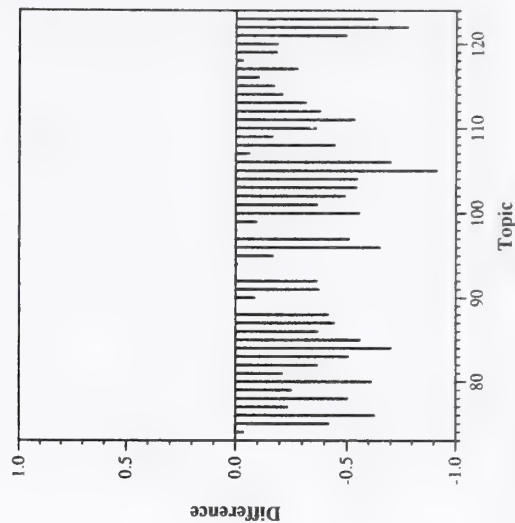
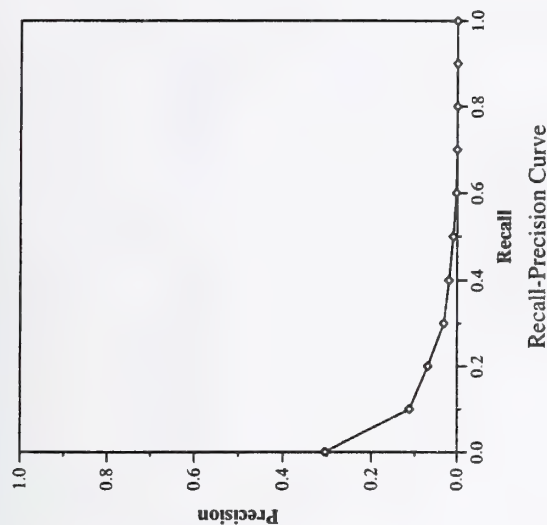


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	tno8b-s1u-limsi
Run Description	unknown, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	480

Recall Level Precision Averages	
Recall	Precision
0.00	0.3053
0.10	0.1127
0.20	0.0697
0.30	0.0317
0.40	0.0195
0.50	0.0095
0.60	0.0021
0.70	0.0008
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0325

Document Level Averages	
	Precision
At 5 docs	0.1061
At 10 docs	0.0898
At 15 docs	0.0667
At 20 docs	0.0633
At 30 docs	0.0490
At 100 docs	0.0267
At 200 docs	0.0202
At 500 docs	0.0144
At 1000 docs	0.0098
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0534

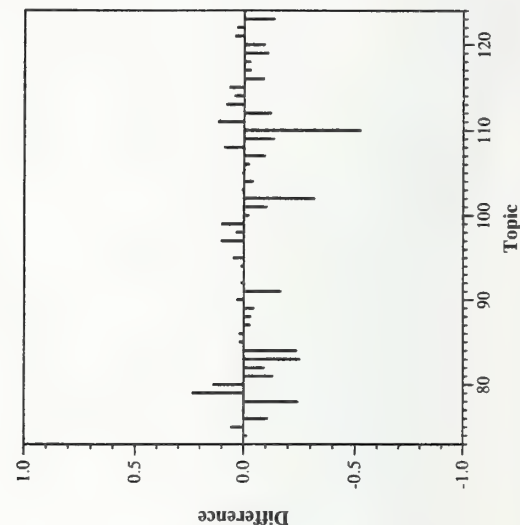
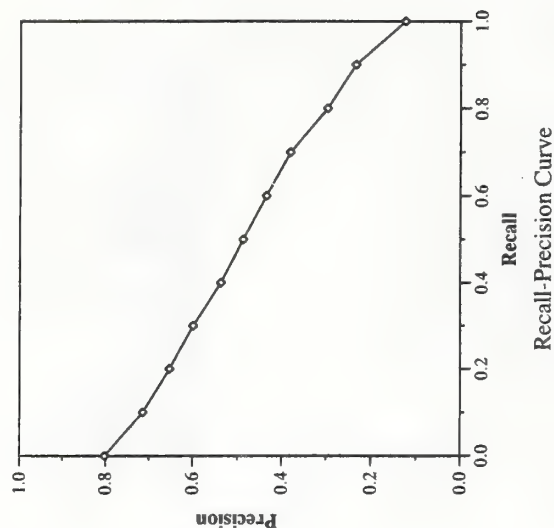


Spoken document retrieval track results — University of Massachusetts

Summary Statistics	
Run Number	umass-bl
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1478

Recall Level Precision Averages	
Recall	Precision
0.00	0.8020
0.10	0.7141
0.20	0.6520
0.30	0.5989
0.40	0.5373
0.50	0.4873
0.60	0.4353
0.70	0.3811
0.80	0.2974
0.90	0.2346
1.00	0.1245
Average precision over all relevant docs	
non-interpolated	0.4677

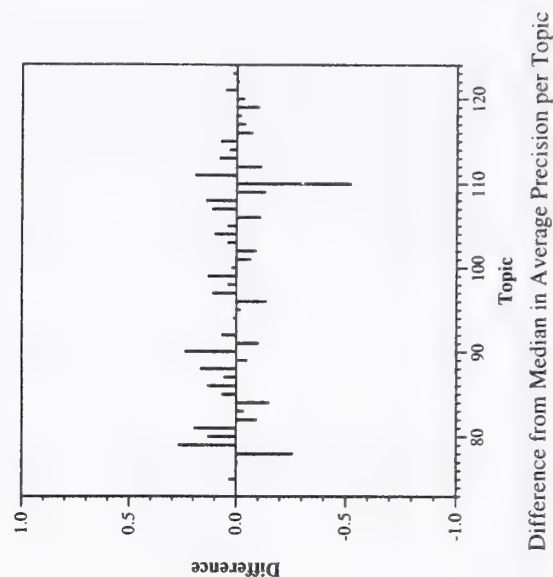
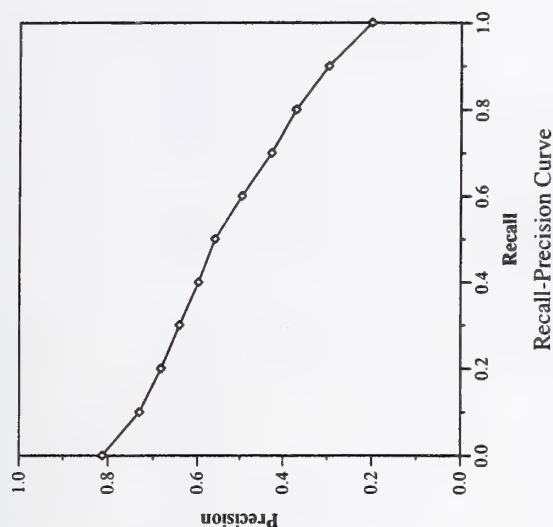
Document Level Averages	
	Precision
At 5 docs	0.6082
At 10 docs	0.5531
At 15 docs	0.5034
At 20 docs	0.4490
At 30 docs	0.3660
At 100 docs	0.1922
At 200 docs	0.1165
At 500 docs	0.0542
At 1000 docs	0.0302
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4691



Summary Statistics	
Run Number	umass-rl
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1580

Recall Level Precision Averages	
Recall	Precision
0.00	0.8123
0.10	0.7310
0.20	0.6832
0.30	0.6414
0.40	0.5981
0.50	0.5601
0.60	0.4961
0.70	0.4272
0.80	0.3713
0.90	0.2975
1.00	0.2009
Average precision over all relevant docs	
non-interpolated	0.5143

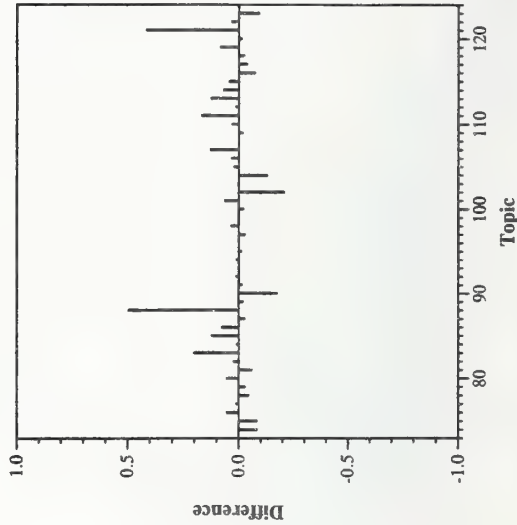
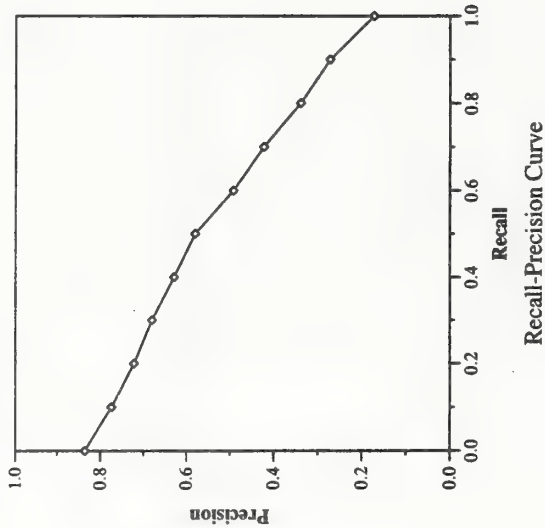
Document Level Averages	
	Precision
At 5 docs	0.6327
At 10 docs	0.5796
At 15 docs	0.5252
At 20 docs	0.4898
At 30 docs	0.3980
At 100 docs	0.2027
At 200 docs	0.1237
At 500 docs	0.0596
At 1000 docs	0.0322
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5127



Summary Statistics	
Run Number	shf-b1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1590

Recall Level Precision Averages	
Recall	Precision
0.00	0.8356
0.10	0.7743
0.20	0.7218
0.30	0.6800
0.40	0.6287
0.50	0.5798
0.60	0.4925
0.70	0.4227
0.80	0.3391
0.90	0.2721
1.00	0.1722
Average precision over all relevant docs	
non-interpolated	0.5298

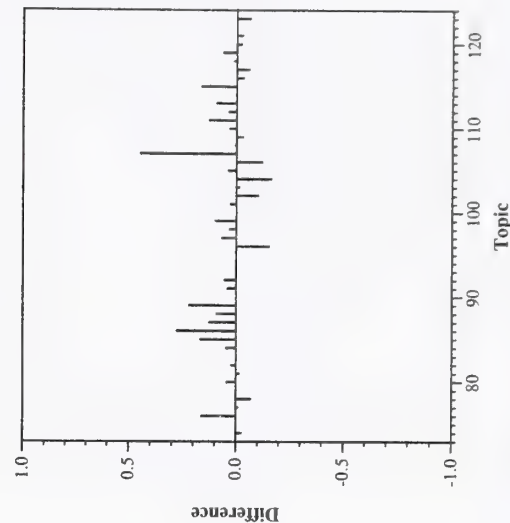
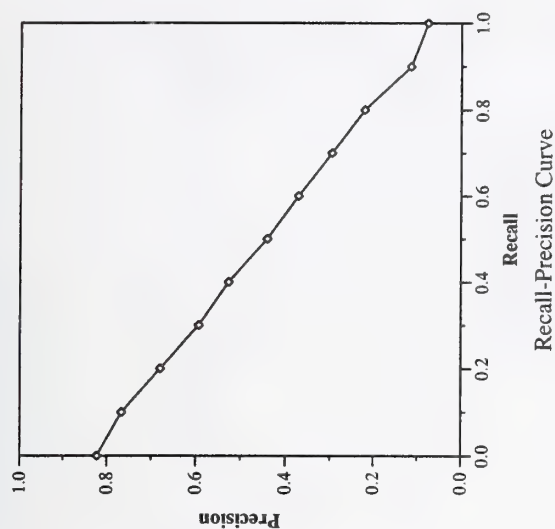
Document Level Averages	
	Precision
At 5 docs	0.7061
At 10 docs	0.6286
At 15 docs	0.5497
At 20 docs	0.4816
At 30 docs	0.4014
At 100 docs	0.2049
At 200 docs	0.1266
At 500 docs	0.0582
At 1000 docs	0.0324
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5101



Summary Statistics		
Run Number	shel-blu	
Run Description	unknown, fixed	
Number of Topics	49	
Total number of documents over all topics		
Retrieved:	49000	
Relevant:	1818	
Rel-ret:	1393	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8221
0.10	0.7673
0.20	0.6796
0.30	0.5934
0.40	0.5264
0.50	0.4405
0.60	0.3718
0.70	0.2964
0.80	0.2212
0.90	0.1152
1.00	0.0768
Average precision over all relevant docs	
non-interpolated	0.4301

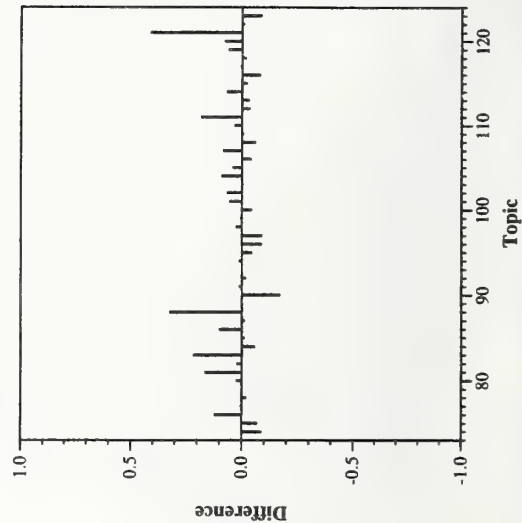
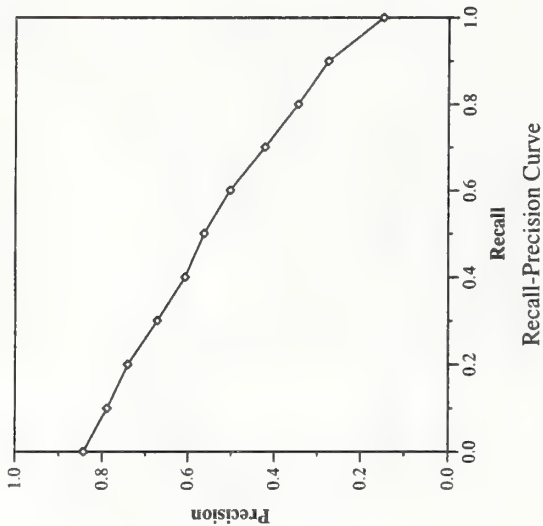
Document Level Averages	
At 5 docs	0.6449
At 10 docs	0.5653
At 15 docs	0.4966
At 20 docs	0.4286
At 30 docs	0.3463
At 100 docs	0.1778
At 200 docs	0.1096
At 500 docs	0.0522
At 1000 docs	0.0284
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4433



Summary Statistics	
Run Number	shef-cr-att-sl
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1622

Recall Level Precision Averages	
Recall	Precision
0.00	0.8435
0.10	0.7892
0.20	0.7416
0.30	0.6721
0.40	0.6070
0.50	0.5627
0.60	0.5029
0.70	0.4228
0.80	0.3455
0.90	0.2754
1.00	0.1479
Average precision over all relevant docs	
non-interpolated	0.5290

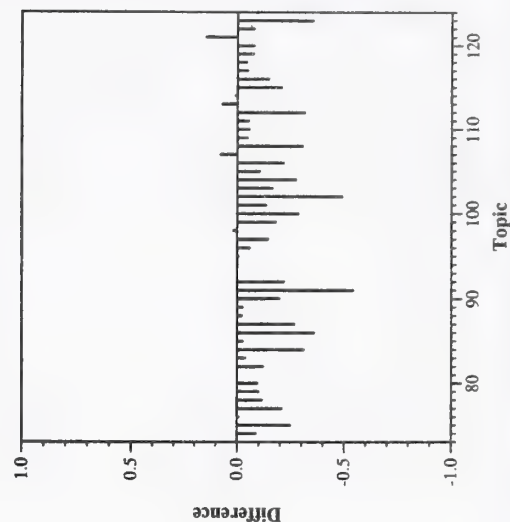
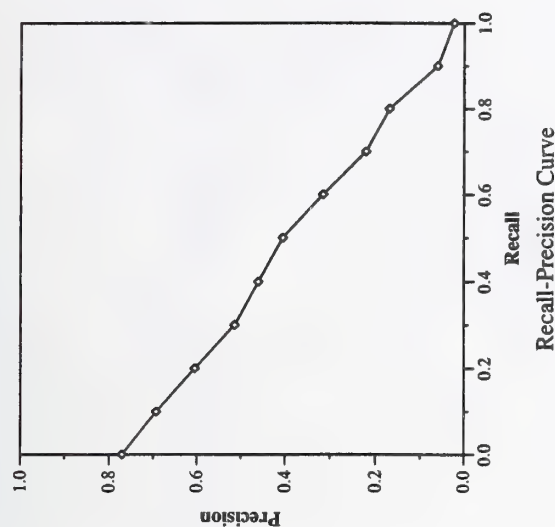
Document Level Averages	
	Precision
At 5 docs	0.7184
At 10 docs	0.6102
At 15 docs	0.5333
At 20 docs	0.4765
At 30 docs	0.3898
At 100 docs	0.2033
At 200 docs	0.1305
At 500 docs	0.0596
At 1000 docs	0.0331
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5130



Summary Statistics	
Run Number	shel-cr-cmu-s1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1299

Recall Level Precision Averages	
Recall	Precision
0.00	0.7706
0.10	0.6934
0.20	0.6047
0.30	0.5142
0.40	0.4604
0.50	0.4053
0.60	0.3158
0.70	0.2211
0.80	0.1692
0.90	0.0607
1.00	0.0237
Average precision over all relevant docs	
non-interpolated	0.3735

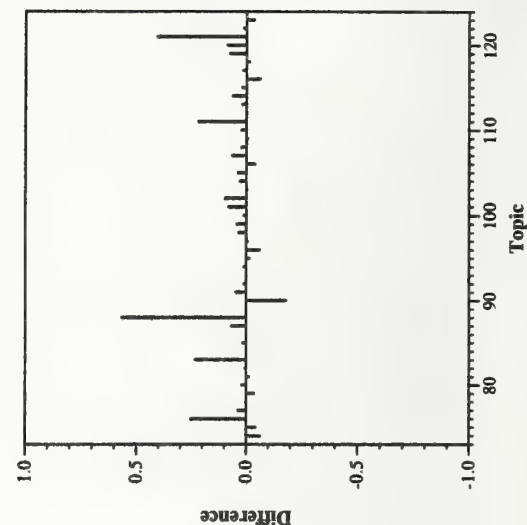
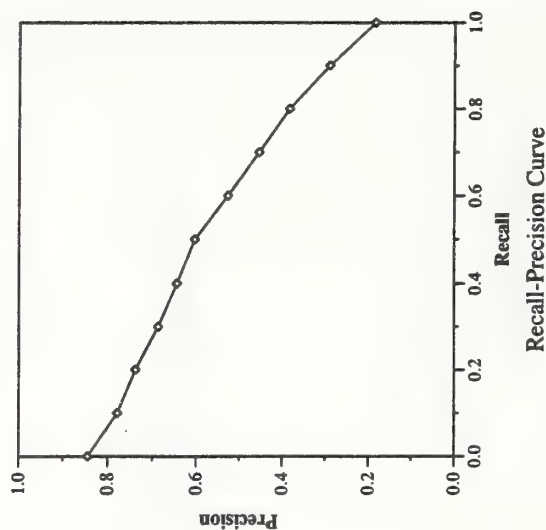
Document Level Averages	
	Precision
At 5 docs	0.5918
At 10 docs	0.4857
At 15 docs	0.4190
At 20 docs	0.3714
At 30 docs	0.3068
At 100 docs	0.1604
At 200 docs	0.0998
At 500 docs	0.0474
At 1000 docs	0.0265
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3931



Summary Statistics	
Run Number	shf-cr-cuhtk-s1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1638

Recall Level Precision Averages	
Recall	Precision
0.00	0.8441
0.10	0.7787
0.20	0.7383
0.30	0.6871
0.40	0.6447
0.50	0.6039
0.60	0.5273
0.70	0.4547
0.80	0.3840
0.90	0.2906
1.00	0.1847
Average precision over all relevant docs	
non-interpolated	0.5484

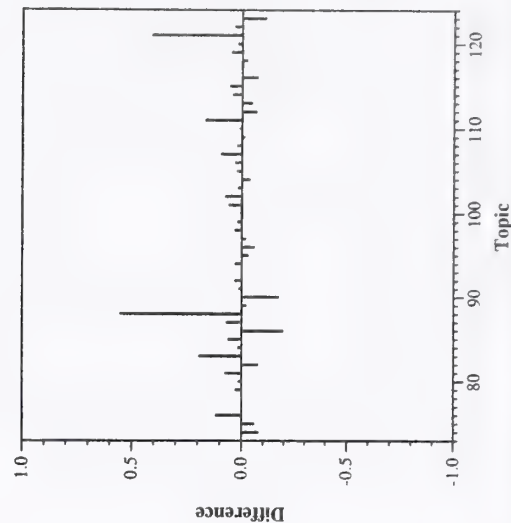
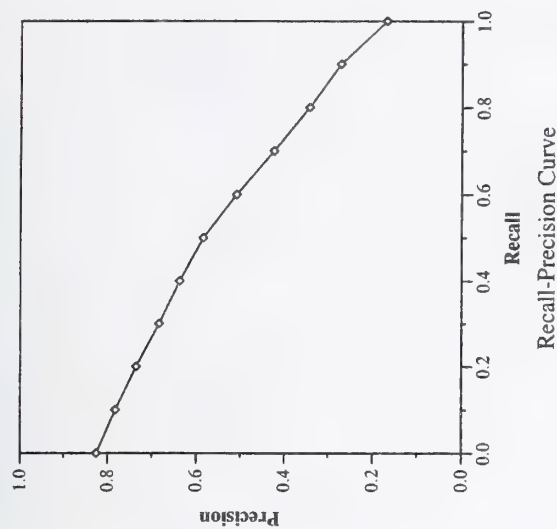
Document Level Averages	
	Precision
At 5 docs	0.7224
At 10 docs	0.6327
At 15 docs	0.5551
At 20 docs	0.4908
At 30 docs	0.4116
At 100 docs	0.2106
At 200 docs	0.1343
At 500 docs	0.0609
At 1000 docs	0.0334
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5174



Summary Statistics	
Run Number	shef-cr-cuhtk-s1p1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1621

Recall Level Precision Averages	
Recall	Precision
0.00	0.8255
0.10	0.7824
0.20	0.7352
0.30	0.6832
0.40	0.6372
0.50	0.5841
0.60	0.5088
0.70	0.4238
0.80	0.3431
0.90	0.2719
1.00	0.1691
Average precision over all relevant docs	
non-interpolated	0.5322

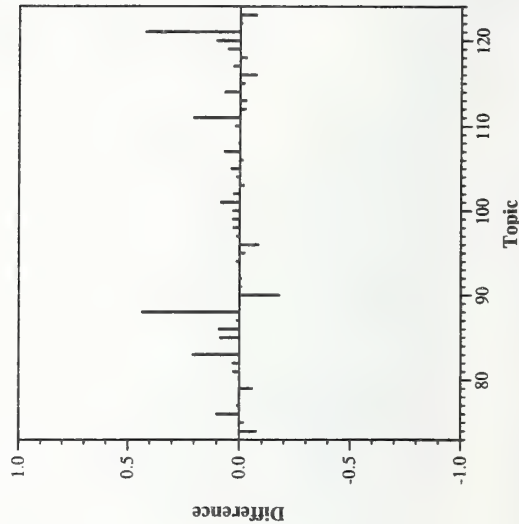
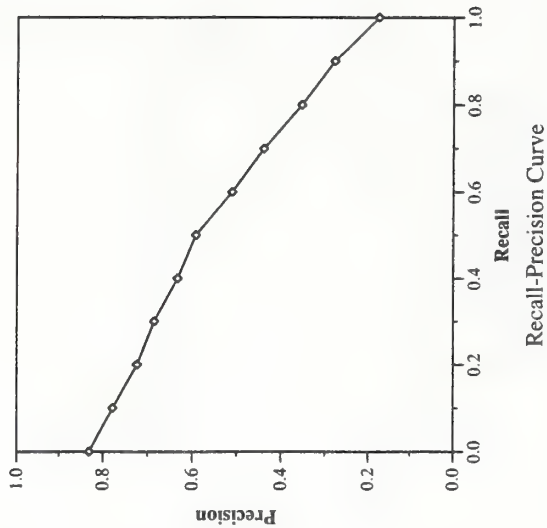
Document Level Averages	
	Precision
At 5 docs	0.7061
At 10 docs	0.6204
At 15 docs	0.5320
At 20 docs	0.4847
At 30 docs	0.3973
At 100 docs	0.2049
At 200 docs	0.1326
At 500 docs	0.0603
At 1000 docs	0.0331
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5113



Summary Statistics	
Run Number	shf-cr-limsi-s1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1613

Recall Level Precision Averages	
Recall	Precision
0.00	0.8307
0.10	0.7781
0.20	0.7239
0.30	0.6859
0.40	0.6334
0.50	0.5925
0.60	0.5112
0.70	0.4397
0.80	0.3542
0.90	0.2788
1.00	0.1763
Average precision over all relevant docs	
non-interpolated	0.5375

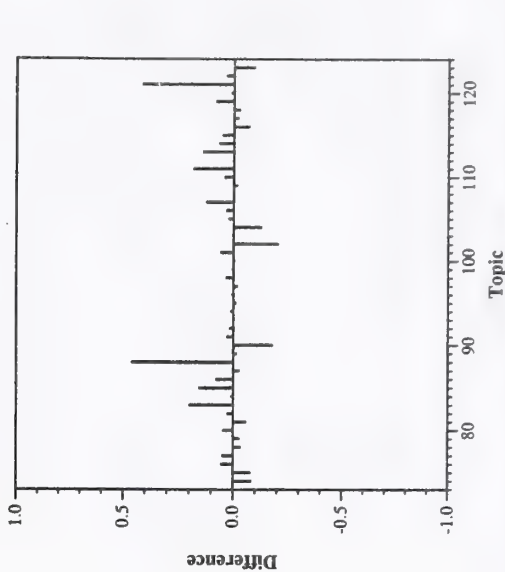
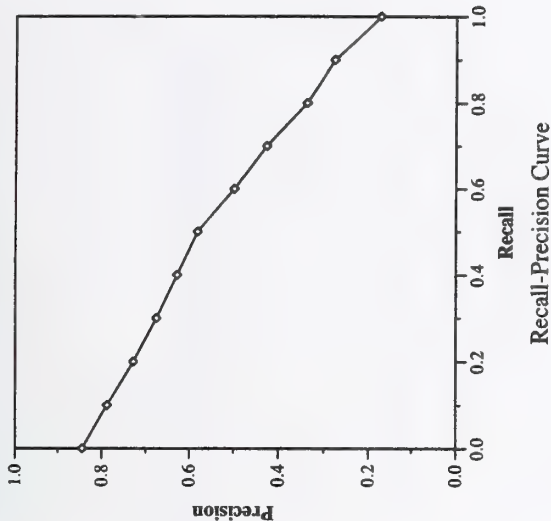
Document Level Averages	
	Precision
At 5 docs	0.7224
At 10 docs	0.6367
At 15 docs	0.5456
At 20 docs	0.4888
At 30 docs	0.4000
At 100 docs	0.2069
At 200 docs	0.1327
At 500 docs	0.0600
At 1000 docs	0.0329
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5222



Summary Statistics	
Run Number	shf-cr-nist-b2
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1587

Recall Level Precision Averages	
Recall	Precision
0.00	0.8441
0.10	0.7882
0.20	0.7285
0.30	0.6760
0.40	0.6283
0.50	0.5811
0.60	0.4988
0.70	0.4263
0.80	0.3380
0.90	0.2768
1.00	0.1728
Average precision over all relevant docs	
non-interpolated	0.5335

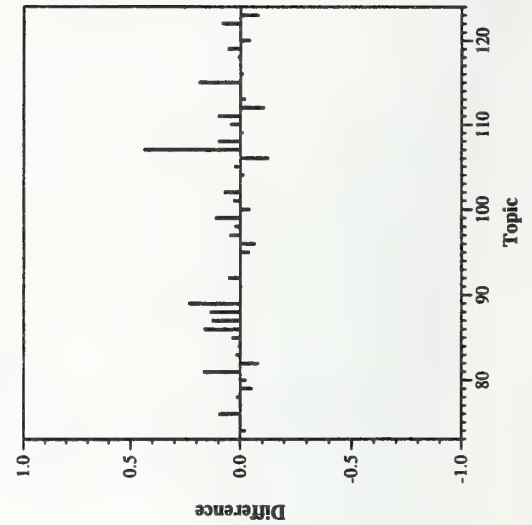
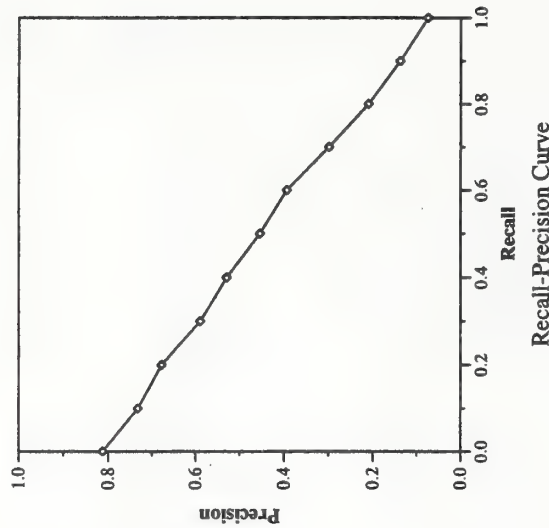
Document Level Averages	
	Precision
At 5 docs	0.7224
At 10 docs	0.6265
At 15 docs	0.5524
At 20 docs	0.4898
At 30 docs	0.4007
At 100 docs	0.2063
At 200 docs	0.1270
At 500 docs	0.0580
At 1000 docs	0.0324
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5068



Summary Statistics	
Run Number	shf-cru-cuhtk-s1plu
Run Description	unknown, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1455

Recall Level Precision Averages	
Recall	Precision
0.00	0.8124
0.10	0.7328
0.20	0.6781
0.30	0.5891
0.40	0.5291
0.50	0.4541
0.60	0.3939
0.70	0.2990
0.80	0.2099
0.90	0.1375
1.00	0.0749
Average precision over all relevant docs	
non-interpolated	0.4311

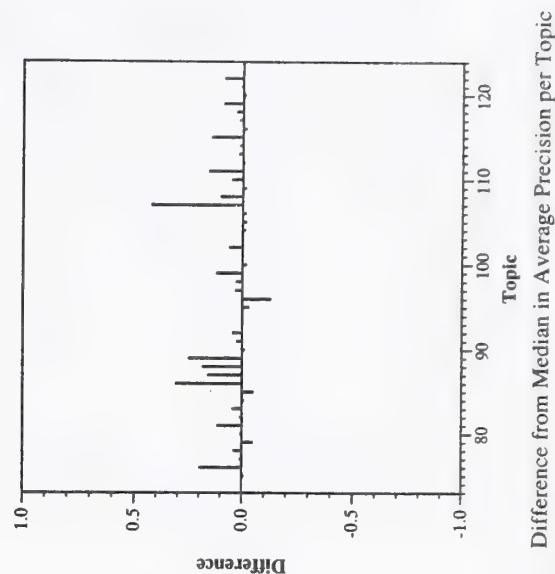
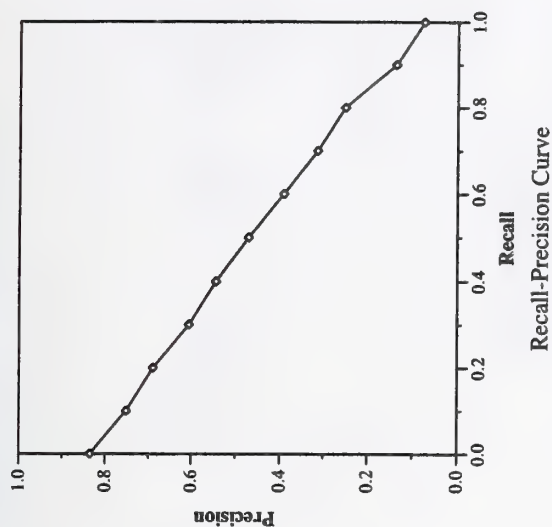
Document Level Averages	
	Precision
At 5 docs	0.6367
At 10 docs	0.5531
At 15 docs	0.4803
At 20 docs	0.4337
At 30 docs	0.3544
At 100 docs	0.1790
At 200 docs	0.1136
At 500 docs	0.0551
At 1000 docs	0.0297
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4465



Summary Statistics	
Run Number	shf-cru-cuhtk-slu
Run Description	unknown, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1458

Recall Level Precision Averages	
Recall	Precision
0.00	0.8345
0.10	0.7518
0.20	0.6898
0.30	0.6054
0.40	0.5433
0.50	0.4707
0.60	0.3918
0.70	0.3146
0.80	0.2519
0.90	0.1372
1.00	0.0752
Average precision over all relevant docs	
non-interpolated	0.4454

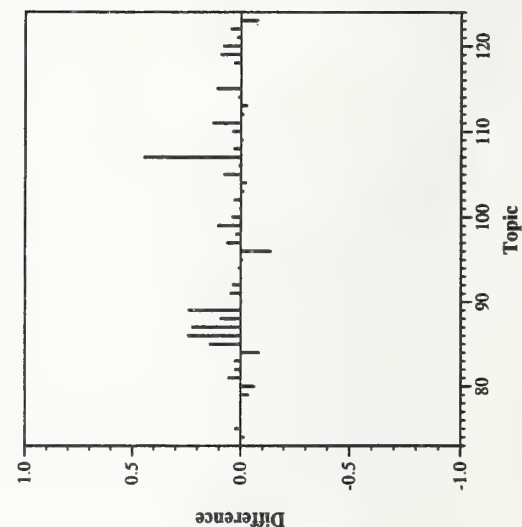
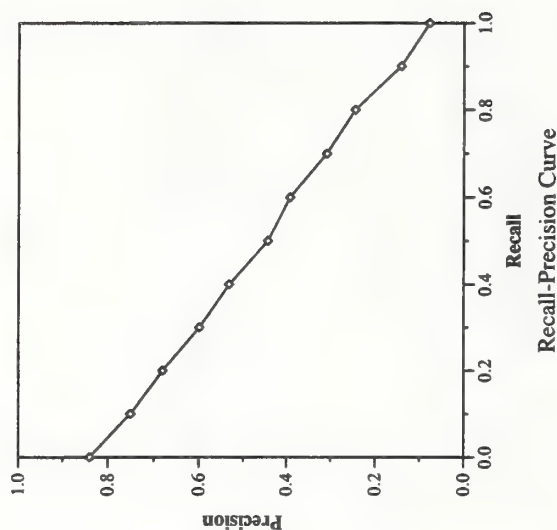
Document Level Averages	
	Precision
At 5 docs	0.6490
At 10 docs	0.5714
At 15 docs	0.4898
At 20 docs	0.4459
At 30 docs	0.3585
At 100 docs	0.1831
At 200 docs	0.1147
At 500 docs	0.0555
At 1000 docs	0.0298
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4489



Summary Statistics	
Run Number	shef-cru-limsi-slu
Run Description	unknown, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1442

Recall Level Precision Averages	
Recall	Precision
0.00	0.8400
0.10	0.7506
0.20	0.6805
0.30	0.5989
0.40	0.5321
0.50	0.4431
0.60	0.3919
0.70	0.3087
0.80	0.2451
0.90	0.1414
1.00	0.0774
Average precision over all relevant docs	
non-interpolated	0.4391

Document Level Averages	
	Precision
At 5 docs	0.6531
At 10 docs	0.5673
At 15 docs	0.5034
At 20 docs	0.4449
At 30 docs	0.3633
At 100 docs	0.1818
At 200 docs	0.1144
At 500 docs	0.0542
At 1000 docs	0.0294
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4410

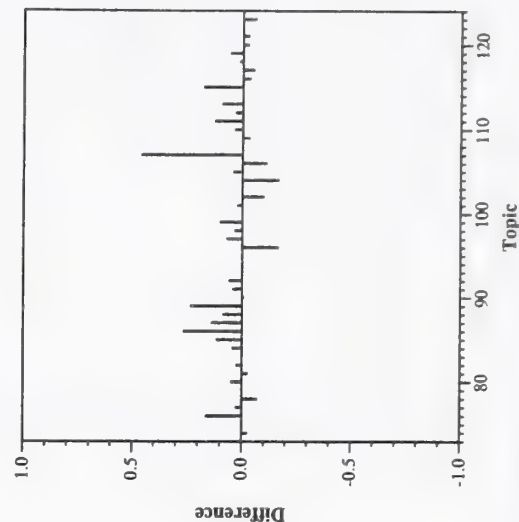
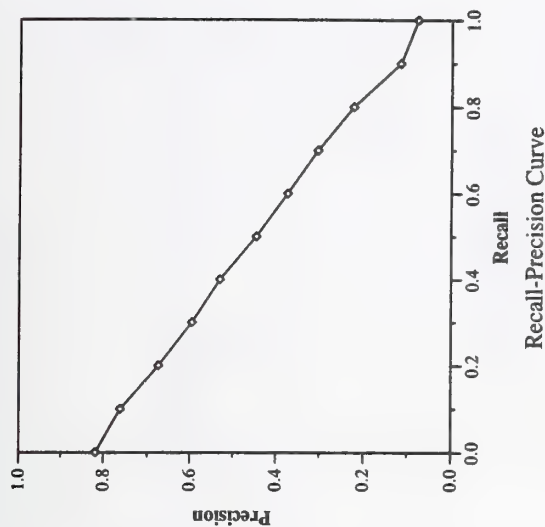


Summary Statistics

Run Number	shf-cru-nist-b2u
Run Description	unknown, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1386

Recall Level Precision Averages	
Recall	Precision
0.00	0.8180
0.10	0.7612
0.20	0.6736
0.30	0.5951
0.40	0.5303
0.50	0.4457
0.60	0.3732
0.70	0.3042
0.80	0.2237
0.90	0.1159
1.00	0.0771
Average precision over all relevant docs	
non-interpolated	0.4299

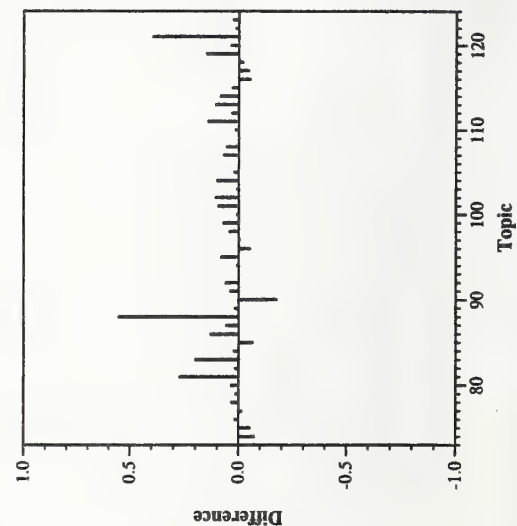
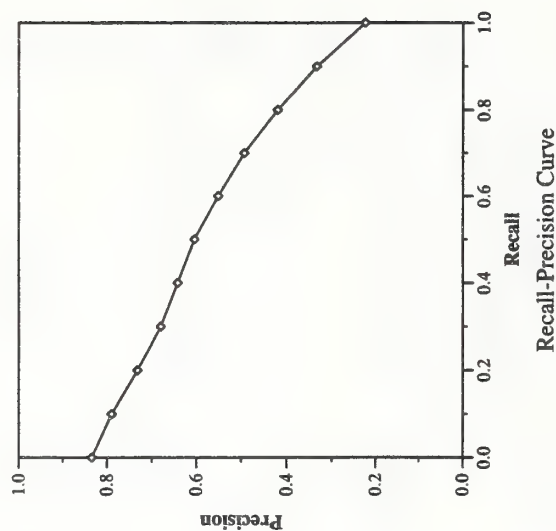
Document Level Averages	
	Precision
At 5 docs	0.6571
At 10 docs	0.5612
At 15 docs	0.4952
At 20 docs	0.4276
At 30 docs	0.3503
At 100 docs	0.1773
At 200 docs	0.1109
At 500 docs	0.0522
At 1000 docs	0.0283
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4448



Summary Statistics	
Run Number	shef-r1
Run Description	known, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1653

Recall Level Precision Averages	
Recall	Precision
0.00	0.8340
0.10	0.7897
0.20	0.7320
0.30	0.6797
0.40	0.6422
0.50	0.6044
0.60	0.5509
0.70	0.4932
0.80	0.4196
0.90	0.3336
1.00	0.2242
Average precision over all relevant docs	
non-interpolated	0.5596

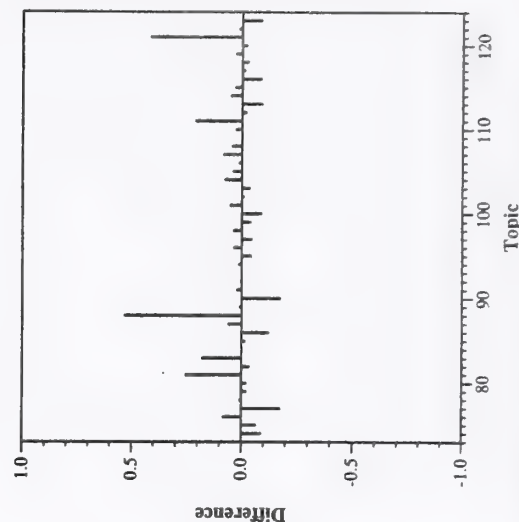
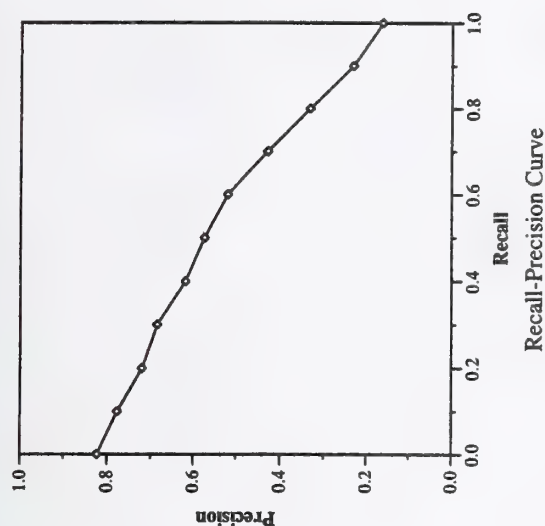
Document Level Averages	
	Precision
At 5 docs	0.7306
At 10 docs	0.6245
At 15 docs	0.5619
At 20 docs	0.5112
At 30 docs	0.4143
At 100 docs	0.2147
At 200 docs	0.1370
At 500 docs	0.0618
At 1000 docs	0.0337
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5424



Summary Statistics		
Run Number	shf-s1	
Run Description	known, fixed	
Number of Topics	49	
Total number of documents over all topics		
Retrieved:	49000	
Relevant:	1818	
Rel-ret:	1594	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8227
0.10	0.7768
0.20	0.7203
0.30	0.6853
0.40	0.6191
0.50	0.5740
0.60	0.5196
0.70	0.4289
0.80	0.3321
0.90	0.2314
1.00	0.1626
Average precision over all relevant docs	
non-interpolated	0.5262

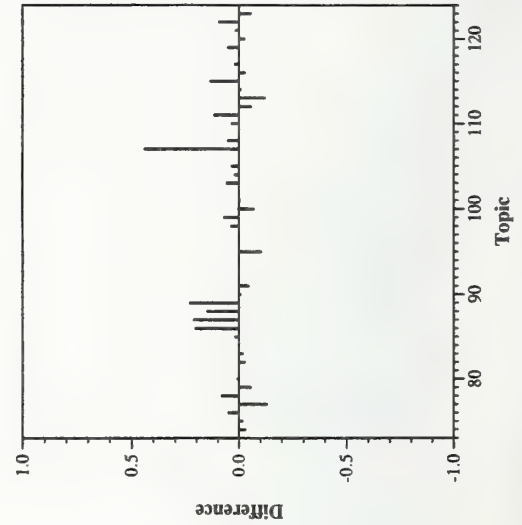
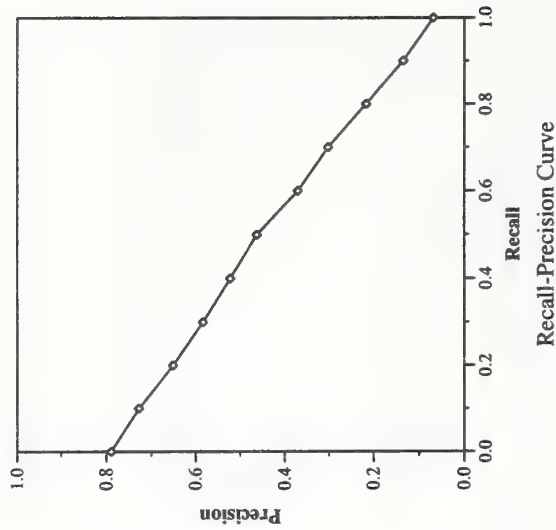
Document Level Averages	
	Precision
At 5 docs	0.6939
At 10 docs	0.6143
At 15 docs	0.5388
At 20 docs	0.4837
At 30 docs	0.3959
At 100 docs	0.2027
At 200 docs	0.1263
At 500 docs	0.0592
At 1000 docs	0.0325
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5110



Summary Statistics	
Run Number	shf-slu
Run Description	unknown, fixed
Number of Topics	49
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1818
Rel-ret:	1393

Recall Level Precision Averages	
Recall	Precision
0.00	0.7897
0.10	0.7277
0.20	0.6507
0.30	0.5833
0.40	0.5225
0.50	0.4629
0.60	0.3706
0.70	0.3020
0.80	0.2163
0.90	0.1348
1.00	0.0691
Average precision over all relevant docs	
non-interpolated	0.4247

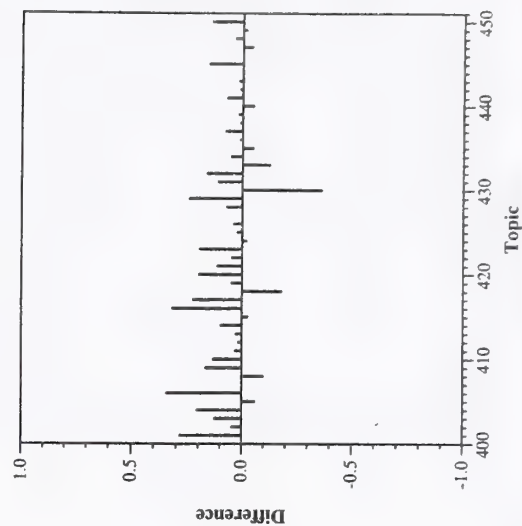
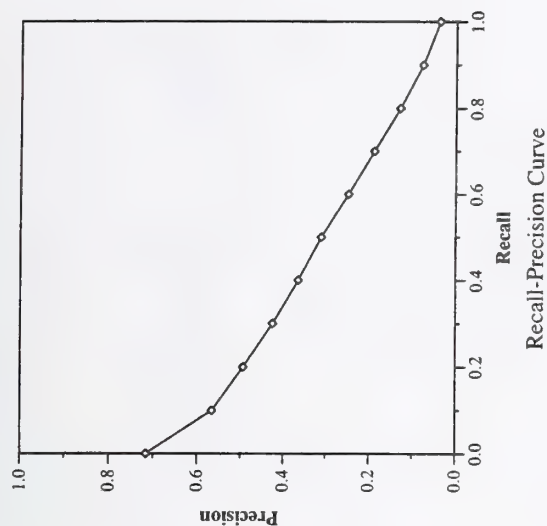
Document Level Averages	
	Precision
At 5 docs	0.6000
At 10 docs	0.5551
At 15 docs	0.4830
At 20 docs	0.4214
At 30 docs	0.3490
At 100 docs	0.1761
At 200 docs	0.1108
At 500 docs	0.0522
At 1000 docs	0.0284
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4431



Summary Statistics	
Run Number	acsys8wm
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1835

Recall Level Precision Averages	
Recall	Precision
0.00	0.7163
0.10	0.5658
0.20	0.4938
0.30	0.4249
0.40	0.3648
0.50	0.3108
0.60	0.2479
0.70	0.1889
0.80	0.1299
0.90	0.0776
1.00	0.0385
Average precision over all relevant docs	
non-interpolated	0.3009

Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4780
At 15 docs	0.4400
At 20 docs	0.3870
At 30 docs	0.3333
At 100 docs	0.1968
At 200 docs	0.1225
At 500 docs	0.0636
At 1000 docs	0.0367
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3221

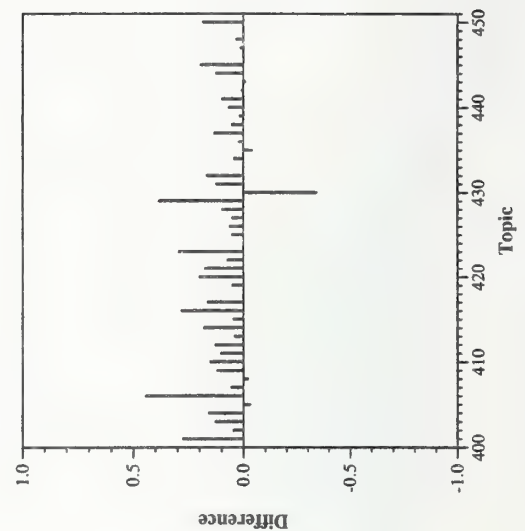
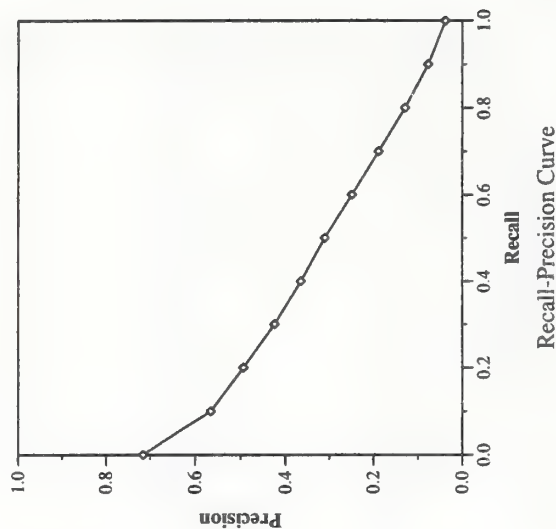


(small) web track results — ACSys

Summary Statistics	
Run Number	acsys8wmp
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1834

Recall Level Precision Averages	
Recall	Precision
0.00	0.7162
0.10	0.5657
0.20	0.4942
0.30	0.4247
0.40	0.3648
0.50	0.3104
0.60	0.2484
0.70	0.1885
0.80	0.1294
0.90	0.0773
1.00	0.0382
Average precision over all relevant docs	
non-interpolated	0.3007

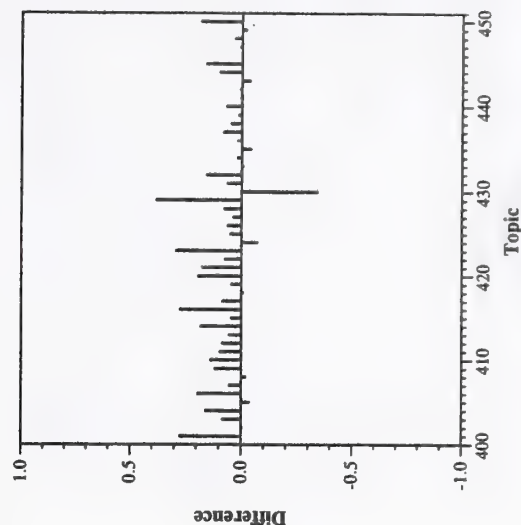
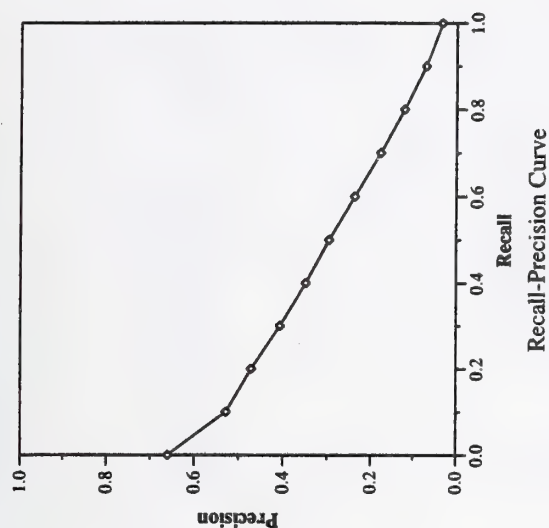
Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4760
At 15 docs	0.4400
At 20 docs	0.3870
At 30 docs	0.3340
At 100 docs	0.1966
At 200 docs	0.1225
At 500 docs	0.0636
At 1000 docs	0.0367
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3217



Summary Statistics	
Run Number	acsys8wmq
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1748

Recall Level Precision Averages	
Recall	Precision
0.00	0.6600
0.10	0.5278
0.20	0.4712
0.30	0.4067
0.40	0.3487
0.50	0.2962
0.60	0.2375
0.70	0.1776
0.80	0.1218
0.90	0.0709
1.00	0.0336
Average precision over all relevant docs	
non-interpolated	0.2804

Document Level Averages	
	Precision
At 5 docs	0.4480
At 10 docs	0.4520
At 15 docs	0.4147
At 20 docs	0.3700
At 30 docs	0.3233
At 100 docs	0.1922
At 200 docs	0.1194
At 500 docs	0.0606
At 1000 docs	0.0350
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3120

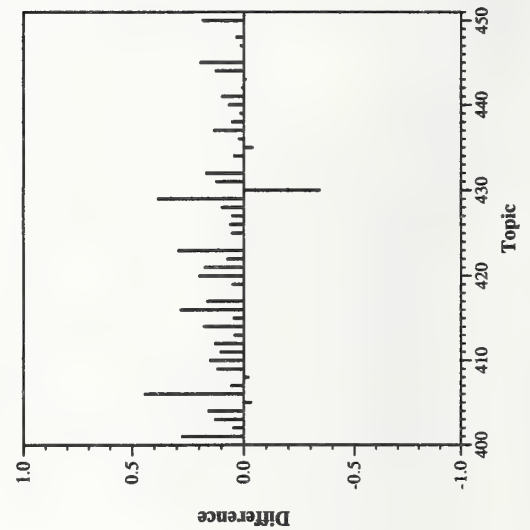
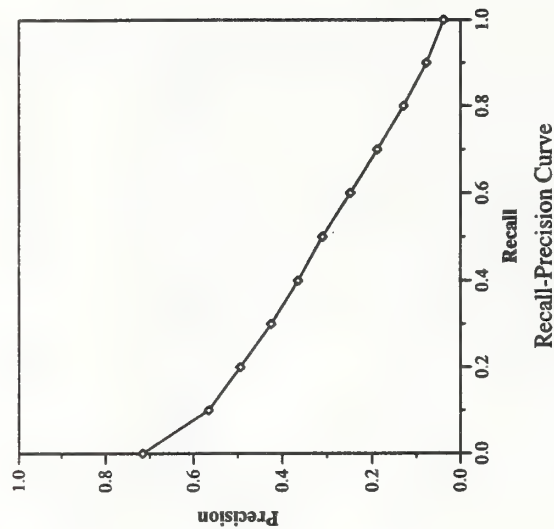


(small) web track results — ACSys

Summary Statistics	
Run Number	acsys8wmr
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1834

Recall Level Precision Averages	
Recall	Precision
0.00	0.7162
0.10	0.5657
0.20	0.4942
0.30	0.4249
0.40	0.3648
0.50	0.3104
0.60	0.2484
0.70	0.1885
0.80	0.1295
0.90	0.0773
1.00	0.0384
Average precision over all relevant docs	
non-interpolated	0.3007

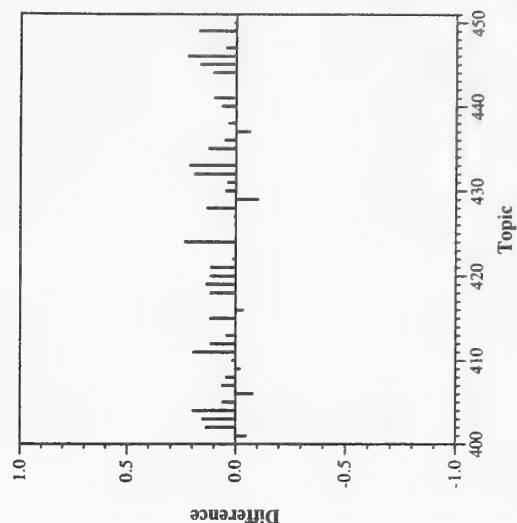
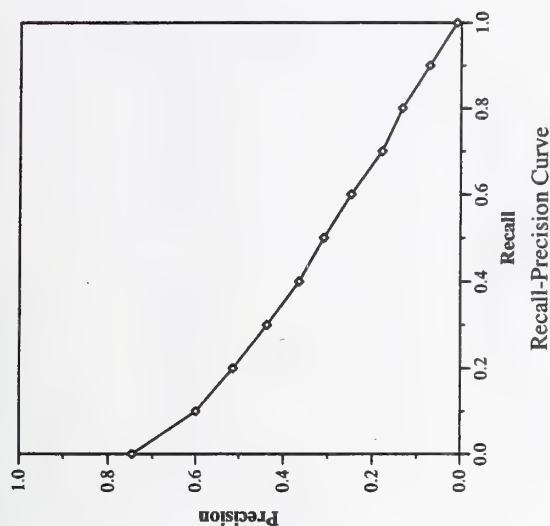
Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4760
At 15 docs	0.4400
At 20 docs	0.3870
At 30 docs	0.3340
At 100 docs	0.1966
At 200 docs	0.1225
At 500 docs	0.0636
At 1000 docs	0.0367
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3217



Summary Statistics	
Run Number	att99wtde
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1872

Recall Level Precision Averages	
Recall	Precision
0.00	0.7467
0.10	0.5996
0.20	0.5142
0.30	0.4371
0.40	0.3651
0.50	0.3092
0.60	0.2484
0.70	0.1792
0.80	0.1337
0.90	0.0720
1.00	0.0101
Average precision over all relevant docs	
non-interpolated	0.3091

Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.4740
At 15 docs	0.4267
At 20 docs	0.3930
At 30 docs	0.3367
At 100 docs	0.2124
At 200 docs	0.1402
At 500 docs	0.0694
At 1000 docs	0.0374
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3264

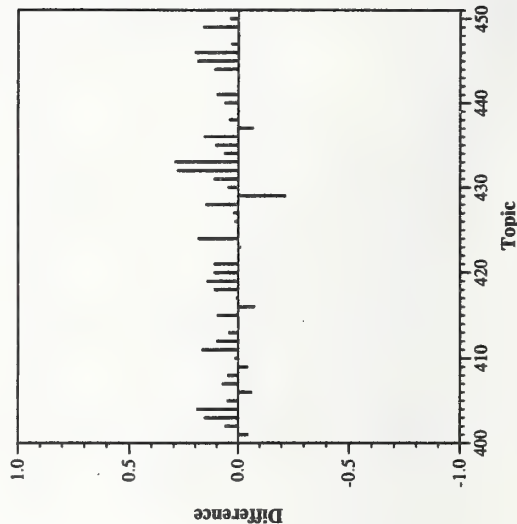
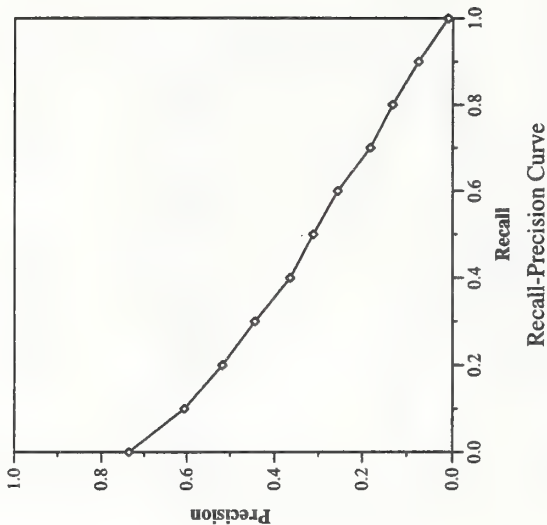


(small) web track results — AT&T Labs Research

Summary Statistics		
Run Number	att99wtde	
Run Description	content-only runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	2279	
Rel-ret:	1888	

Recall Level Precision Averages	
Recall	Precision
0.00	0.7361
0.10	0.6060
0.20	0.5179
0.30	0.4447
0.40	0.3646
0.50	0.3117
0.60	0.2561
0.70	0.1816
0.80	0.1320
0.90	0.0747
1.00	0.0102
Average precision over all relevant docs	
non-interpolated	0.3113

Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.4700
At 15 docs	0.4440
At 20 docs	0.4110
At 30 docs	0.3500
At 100 docs	0.2106
At 200 docs	0.1427
At 500 docs	0.0700
At 1000 docs	0.0378
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3329

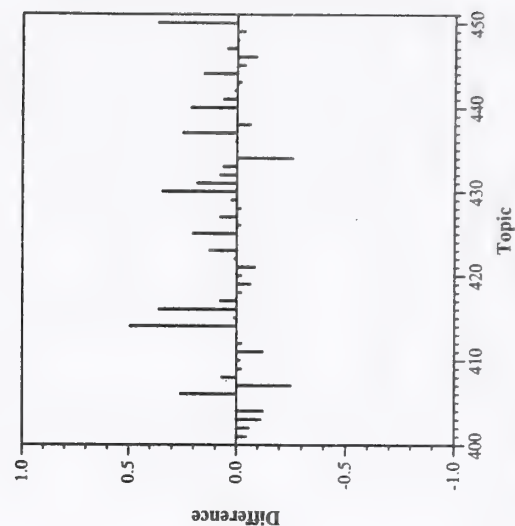
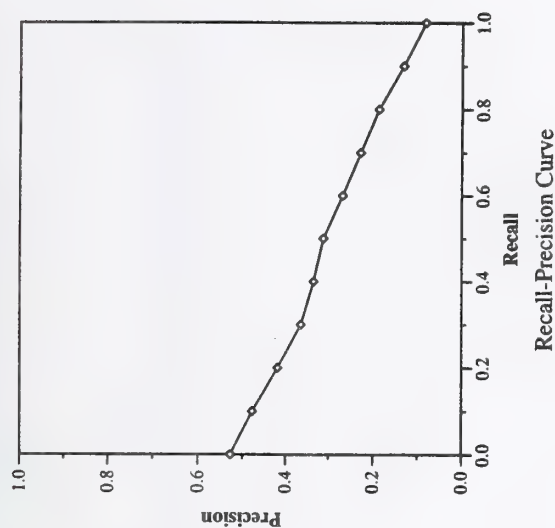


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	CL99WebH
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1933

Recall Level Precision Averages	
Recall	Precision
0.00	0.5275
0.10	0.4776
0.20	0.4187
0.30	0.3636
0.40	0.3341
0.50	0.3114
0.60	0.2681
0.70	0.2278
0.80	0.1873
0.90	0.1317
1.00	0.0827
Average precision over all relevant docs	
non-interpolated	0.2845

Document Level Averages	
	Precision
At 5 docs	0.3640
At 10 docs	0.3180
At 15 docs	0.2987
At 20 docs	0.2830
At 30 docs	0.2580
At 100 docs	0.1804
At 200 docs	0.1279
At 500 docs	0.0690
At 1000 docs	0.0387
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2837

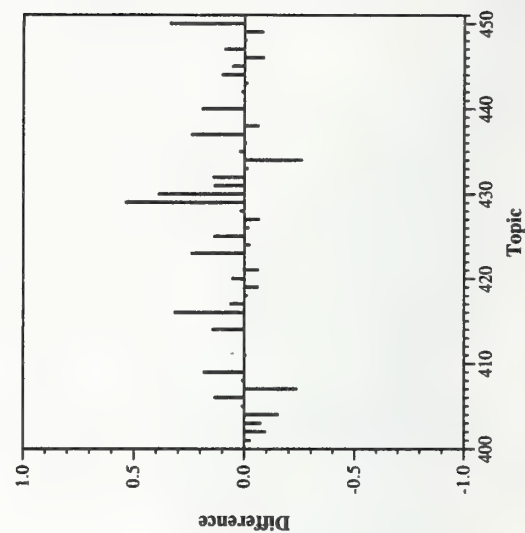
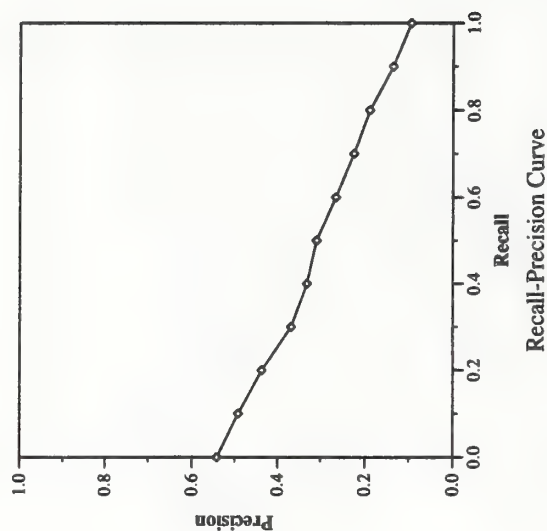


(small) web track results — CLARITECH Corporation

Summary Statistics	
Run Number	CL99WebM
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1924

Recall Level Precision Averages	
Recall	Precision
0.00	0.5419
0.10	0.4920
0.20	0.4367
0.30	0.3682
0.40	0.3314
0.50	0.3090
0.60	0.2653
0.70	0.2249
0.80	0.1893
0.90	0.1361
1.00	0.0955
Average precision over all relevant docs	
non-interpolated	0.2889

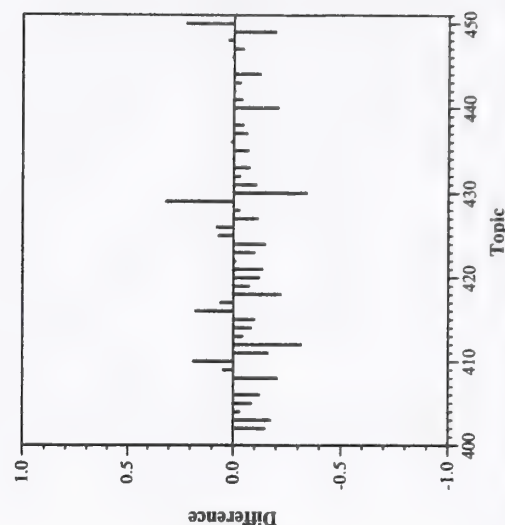
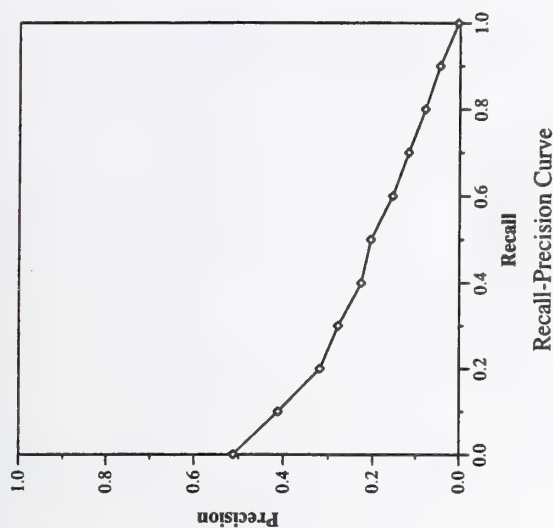
Document Level Averages	
	Precision
At 5 docs	0.3560
At 10 docs	0.3260
At 15 docs	0.3040
At 20 docs	0.2880
At 30 docs	0.2647
At 100 docs	0.1724
At 200 docs	0.1235
At 500 docs	0.0687
At 1000 docs	0.0385
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2960



Summary Statistics	
Run Number	DCU99C01
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	8314
Relevant:	2279
Rel-ret:	1017

Recall Level Precision Averages	
Recall	Precision
0.00	0.5119
0.10	0.4117
0.20	0.3190
0.30	0.2783
0.40	0.2265
0.50	0.2050
0.60	0.1549
0.70	0.1175
0.80	0.0790
0.90	0.0452
1.00	0.0040
Average precision over all relevant docs	
non-interpolated	0.1936

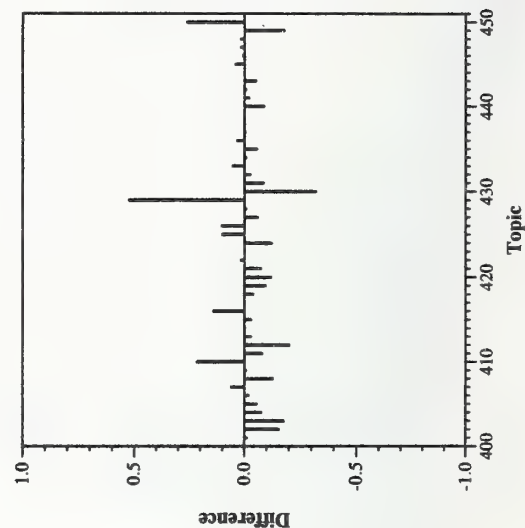
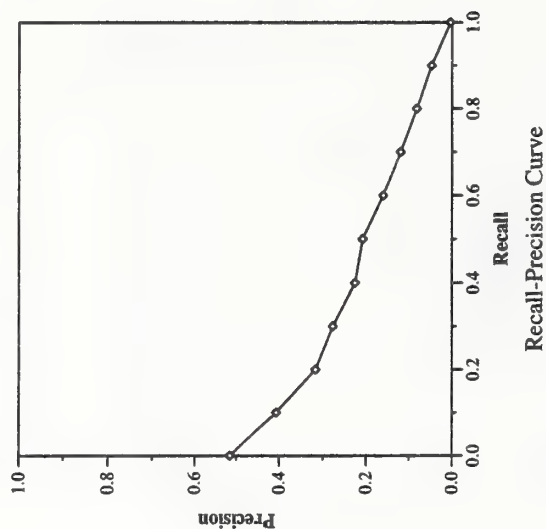
Document Level Averages	
	Precision
At 5 docs	0.3440
At 10 docs	0.3100
At 15 docs	0.2733
At 20 docs	0.2510
At 30 docs	0.2267
At 100 docs	0.1506
At 200 docs	0.1017
At 500 docs	0.0407
At 1000 docs	0.0203
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2300



Summary Statistics	
Run Number	DCU99L01
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	8314
Relevant:	2279
Rel-ret:	1017

Recall Level Precision Averages	
Recall	Precision
0.00	0.5151
0.10	0.4069
0.20	0.3170
0.30	0.2769
0.40	0.2255
0.50	0.2077
0.60	0.1599
0.70	0.1194
0.80	0.0819
0.90	0.0473
1.00	0.0039
Average precision over all relevant docs	
non-interpolated	0.1939

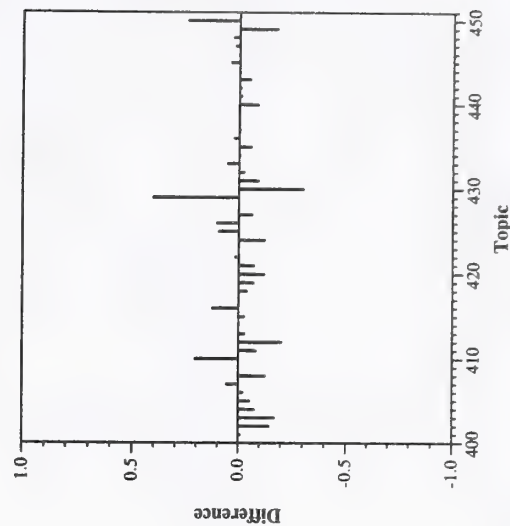
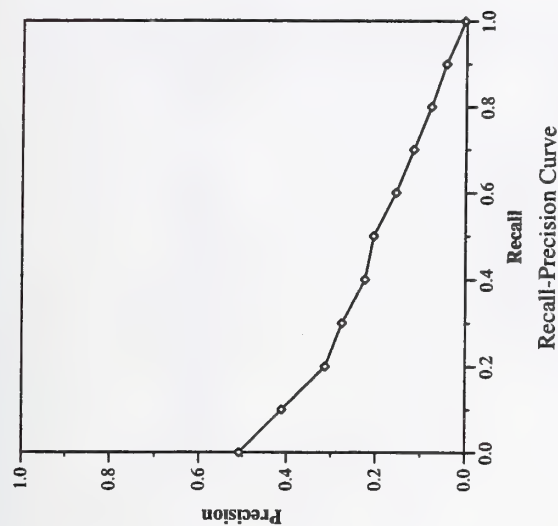
Document Level Averages	
At 5 docs	0.3400
At 10 docs	0.3140
At 15 docs	0.2693
At 20 docs	0.2490
At 30 docs	0.2260
At 100 docs	0.1500
At 200 docs	0.1017
At 500 docs	0.0407
At 1000 docs	0.0203
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2358



Summary Statistics	
Run Number	DCU99L02
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	8314
Relevant:	2279
Rel-ret:	1017

Recall Level Precision Averages	
Recall	Precision
0.00	0.5078
0.10	0.4110
0.20	0.3134
0.30	0.2755
0.40	0.2242
0.50	0.2051
0.60	0.1554
0.70	0.1166
0.80	0.0783
0.90	0.0450
1.00	0.0040
Average precision over all relevant docs	
non-interpolated	0.1921

Document Level Averages	
At 5 docs	0.3320
At 10 docs	0.3060
At 15 docs	0.2693
At 20 docs	0.2510
At 30 docs	0.2287
At 100 docs	0.1506
At 200 docs	0.1017
At 500 docs	0.0407
At 1000 docs	0.0203
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2326

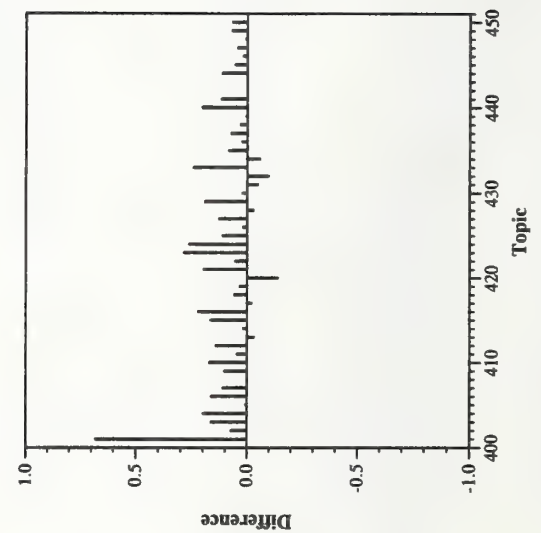
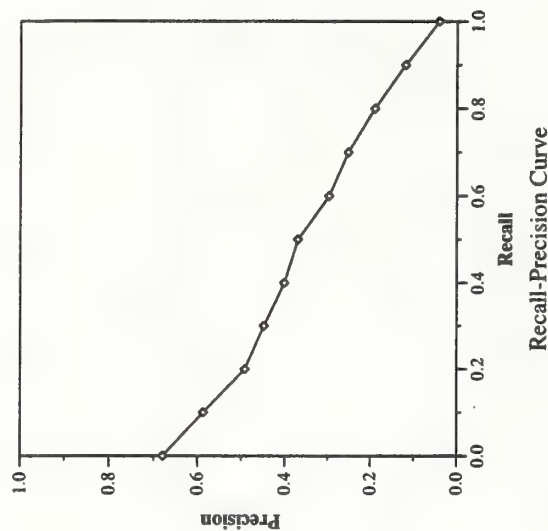


(small) web track results — Fujitsu Laboratories. Ltd.

Summary Statistics	
Run Number	Flab8wtdN
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1954

Recall Level Precision Averages	
Recall	Precision
0.00	0.6790
0.10	0.5869
0.20	0.4919
0.30	0.4483
0.40	0.4011
0.50	0.3698
0.60	0.2973
0.70	0.2526
0.80	0.1902
0.90	0.1190
1.00	0.0409
Average precision over all relevant docs	
non-interpolated	0.3320

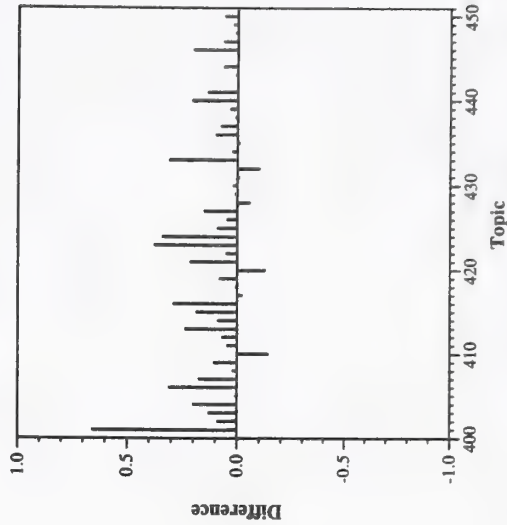
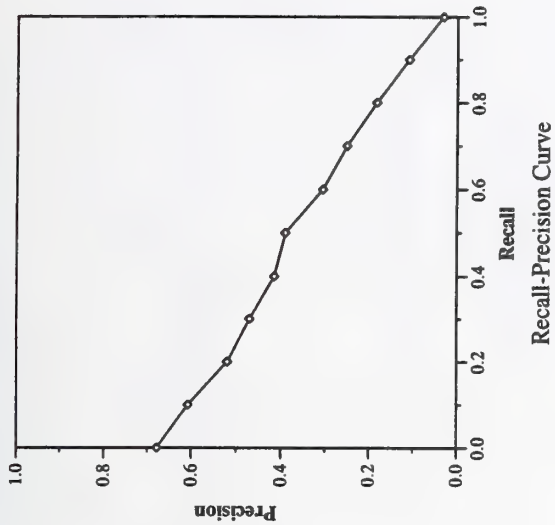
Document Level Averages	
	Precision
At 5 docs	0.4640
At 10 docs	0.4360
At 15 docs	0.4280
At 20 docs	0.3980
At 30 docs	0.3553
At 100 docs	0.2104
At 200 docs	0.1439
At 500 docs	0.0718
At 1000 docs	0.0391
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3552



Summary Statistics	
Run Number	Flab8wtdnN
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1988

Recall Level Precision Averages	
Recall	Precision
0.00	0.6775
0.10	0.6073
0.20	0.5197
0.30	0.4705
0.40	0.4155
0.50	0.3907
0.60	0.3059
0.70	0.2507
0.80	0.1832
0.90	0.1100
1.00	0.0325
Average precision over all relevant docs	
non-interpolated	0.3405

Document Level Averages	
	Precision
At 5 docs	0.4920
At 10 docs	0.4640
At 15 docs	0.4307
At 20 docs	0.4010
At 30 docs	0.3600
At 100 docs	0.2126
At 200 docs	0.1465
At 500 docs	0.0729
At 1000 docs	0.0398
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3535



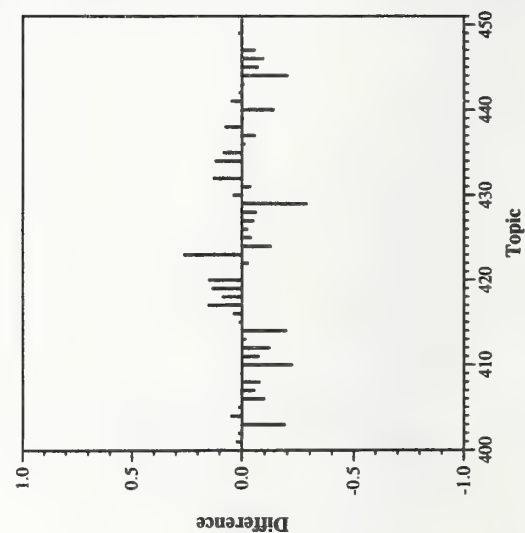
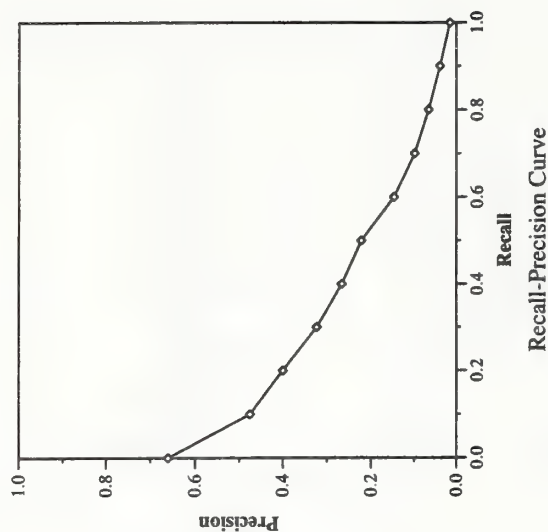
Difference from Median in Average Precision per Topic

(small) web track results — IIT/AAT/NCR

Summary Statistics		
Run Number	iit99wt1	
Run Description	content-only runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	2279	
Rel-ret:	1575	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6605
0.10	0.4752
0.20	0.4000
0.30	0.3227
0.40	0.2660
0.50	0.2222
0.60	0.1474
0.70	0.0990
0.80	0.0667
0.90	0.0391
1.00	0.0156
Average precision over all relevant docs	
non-interpolated	0.2265

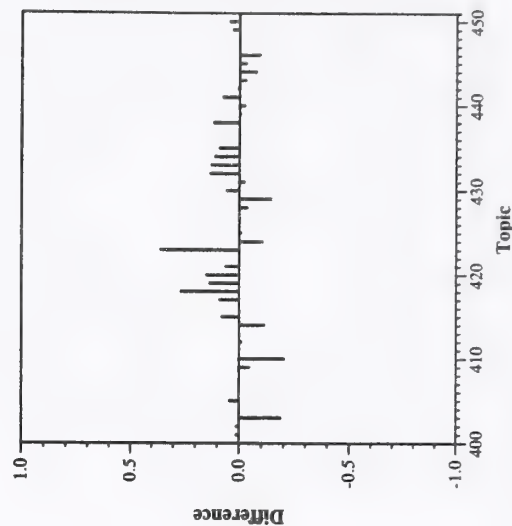
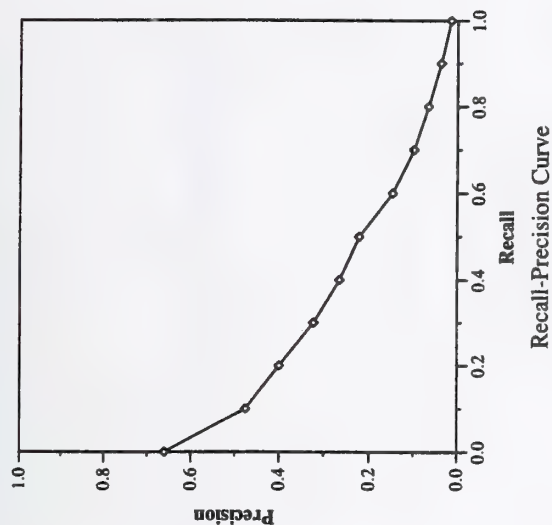
Document Level Averages	
	Precision
At 5 docs	0.4360
At 10 docs	0.4100
At 15 docs	0.3453
At 20 docs	0.3150
At 30 docs	0.2740
At 100 docs	0.1644
At 200 docs	0.1072
At 500 docs	0.0550
At 1000 docs	0.0315
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2727



Summary Statistics	
Run Number	iit99wt2
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1572

Recall Level Precision Averages	
Recall	Precision
0.00	0.6605
0.10	0.4752
0.20	0.4000
0.30	0.3225
0.40	0.2655
0.50	0.2220
0.60	0.1474
0.70	0.0990
0.80	0.0663
0.90	0.0373
1.00	0.0156
Average precision over all relevant docs	
non-interpolated	0.2265

Document Level Averages	
	Precision
At 5 docs	0.4360
At 10 docs	0.4100
At 15 docs	0.3453
At 20 docs	0.3150
At 30 docs	0.2740
At 100 docs	0.1644
At 200 docs	0.1072
At 500 docs	0.0550
At 1000 docs	0.0314
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2727

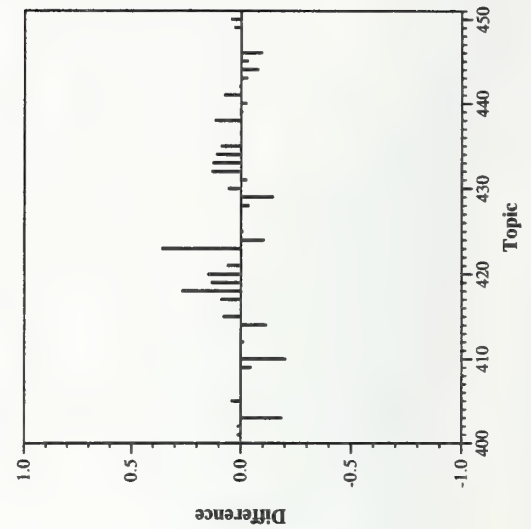
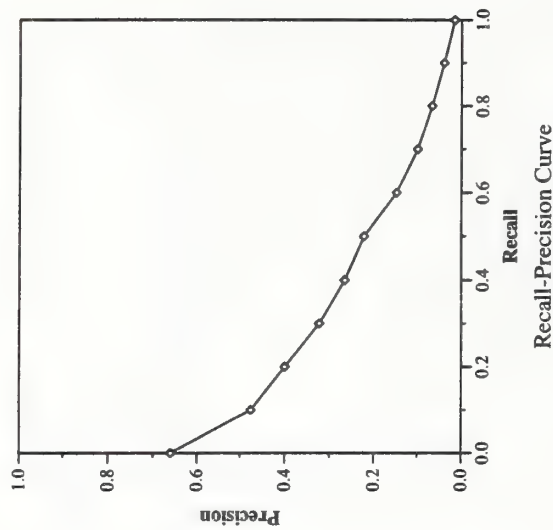


(small) web track results — IIT/AAT/NCR

Summary Statistics		
Run Number	iit99wt3	
Run Description	content-link runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	2279	
Rel-ret:	1568	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6605
0.10	0.4752
0.20	0.4000
0.30	0.3225
0.40	0.2655
0.50	0.2220
0.60	0.1480
0.70	0.0990
0.80	0.0663
0.90	0.0391
1.00	0.0156
Average precision over all relevant docs	
non-interpolated	0.2264

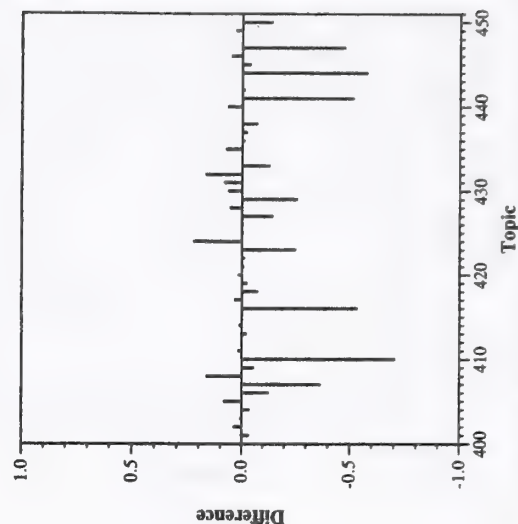
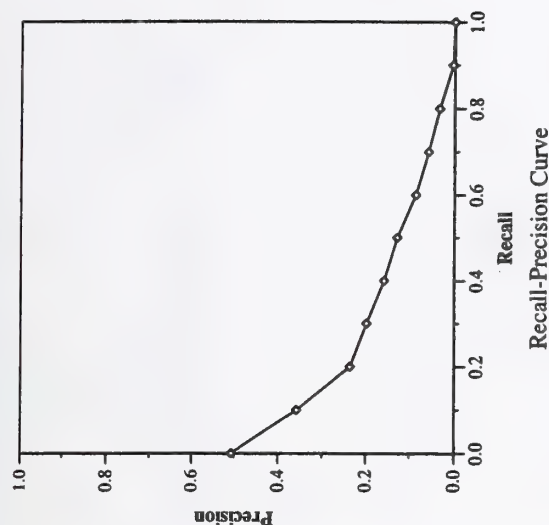
Document Level Averages	
	Precision
At 5 docs	0.4360
At 10 docs	0.4100
At 15 docs	0.3453
At 20 docs	0.3150
At 30 docs	0.2740
At 100 docs	0.1644
At 200 docs	0.1072
At 500 docs	0.0550
At 1000 docs	0.0314
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2727



Summary Statistics	
Run Number	Mer8Wci1
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1338

Recall Level Precision Averages	
Recall	Precision
0.00	0.5063
0.10	0.3581
0.20	0.2358
0.30	0.1980
0.40	0.1575
0.50	0.1276
0.60	0.0858
0.70	0.0577
0.80	0.0329
0.90	0.0038
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1401

Document Level Averages	
	Precision
At 5 docs	0.3280
At 10 docs	0.2840
At 15 docs	0.2293
At 20 docs	0.1950
At 30 docs	0.1740
At 100 docs	0.1236
At 200 docs	0.0876
At 500 docs	0.0465
At 1000 docs	0.0268
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1666

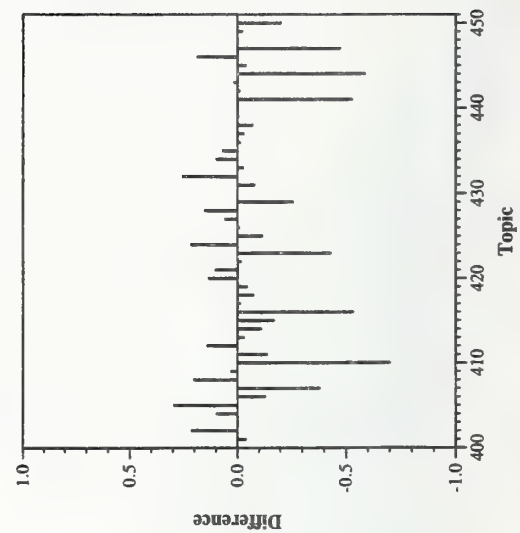
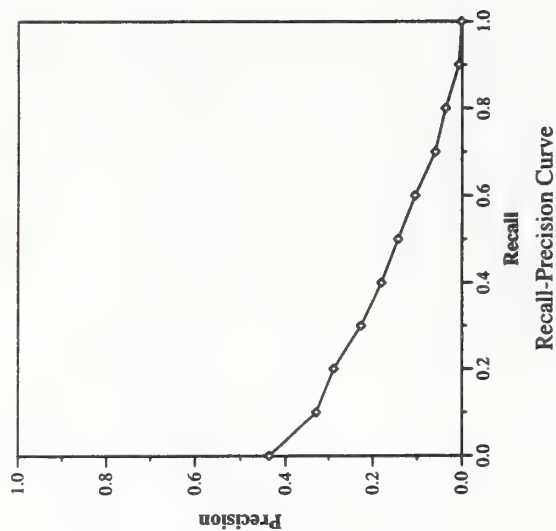


(small) web track results — IRIT/SIG

Summary Statistics	
Run Number	Mer8Wci2
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1258

Recall Level Precision Averages	
Recall	Precision
0.00	0.4372
0.10	0.3297
0.20	0.2890
0.30	0.2266
0.40	0.1806
0.50	0.1428
0.60	0.1049
0.70	0.0600
0.80	0.0371
0.90	0.0075
1.00	0.0011
Average precision over all relevant docs	
non-interpolated	0.1488

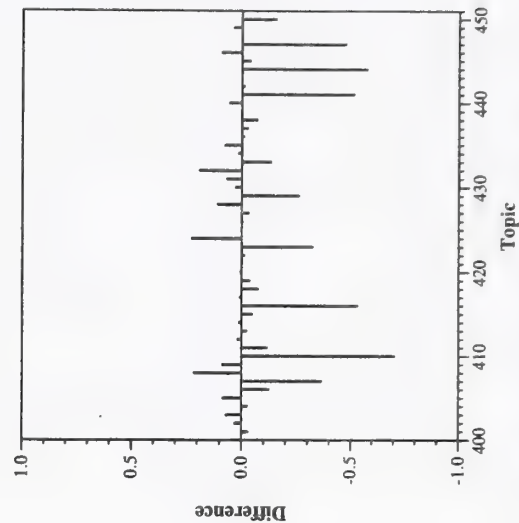
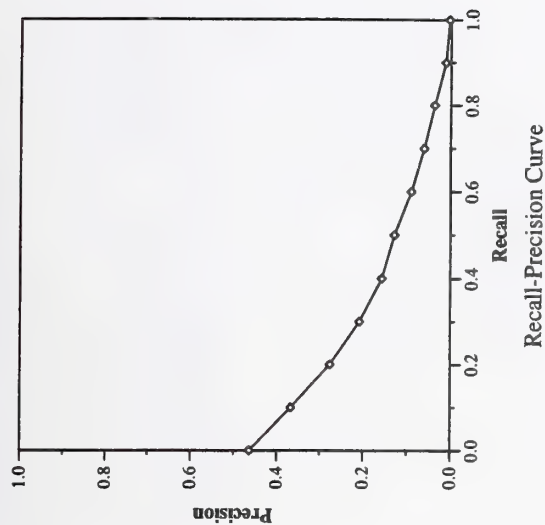
Document Level Averages	
	Precision
At 5 docs	0.2680
At 10 docs	0.2220
At 15 docs	0.2200
At 20 docs	0.2160
At 30 docs	0.2047
At 100 docs	0.1352
At 200 docs	0.0879
At 500 docs	0.0438
At 1000 docs	0.0252
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1771



Summary Statistics	
Run Number	Mer8Wci3
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1352

Recall Level Precision Averages	
Recall	Precision
0.00	0.4634
0.10	0.3673
0.20	0.2763
0.30	0.2079
0.40	0.1565
0.50	0.1282
0.60	0.0895
0.70	0.0604
0.80	0.0365
0.90	0.0119
1.00	0.0036
Average precision over all relevant docs	
non-interpolated	0.1435

Document Level Averages	
	Precision
At 5 docs	0.2520
At 10 docs	0.2620
At 15 docs	0.2427
At 20 docs	0.2130
At 30 docs	0.1900
At 100 docs	0.1282
At 200 docs	0.0892
At 500 docs	0.0471
At 1000 docs	0.0270
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1741

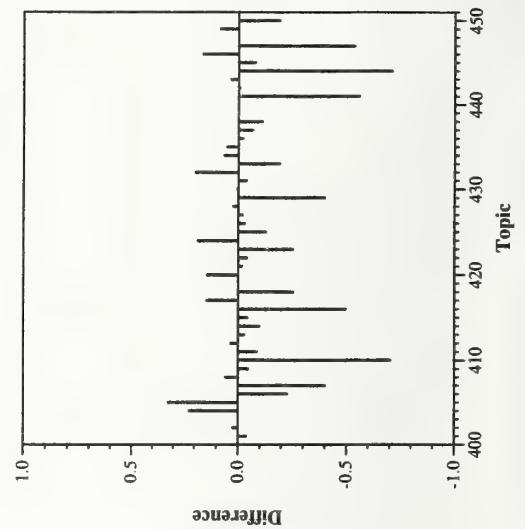
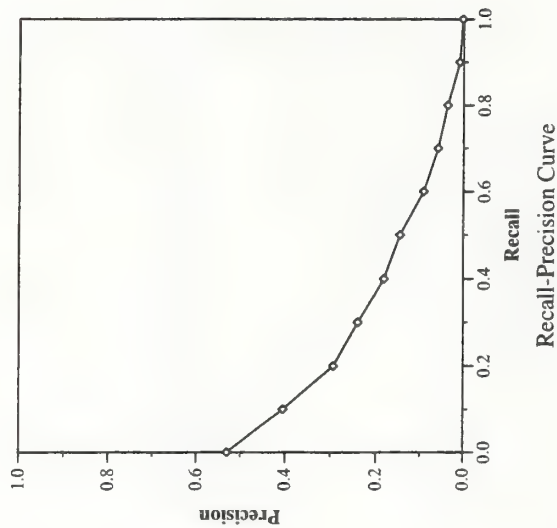


(small) web track results — IRIT/SIG

Summary Statistics	
Run Number	Mer8Wctd
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1286

Recall Level Precision Averages	
Recall	Precision
0.00	0.5301
0.10	0.4046
0.20	0.2938
0.30	0.2404
0.40	0.1822
0.50	0.1460
0.60	0.0920
0.70	0.0590
0.80	0.0370
0.90	0.0093
1.00	0.0020
Average precision over all relevant docs	
non-interpolated	0.1638

Document Level Averages	
	Precision
At 5 docs	0.3320
At 10 docs	0.3020
At 15 docs	0.2787
At 20 docs	0.2430
At 30 docs	0.2153
At 100 docs	0.1410
At 200 docs	0.0881
At 500 docs	0.0443
At 1000 docs	0.0257
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1957

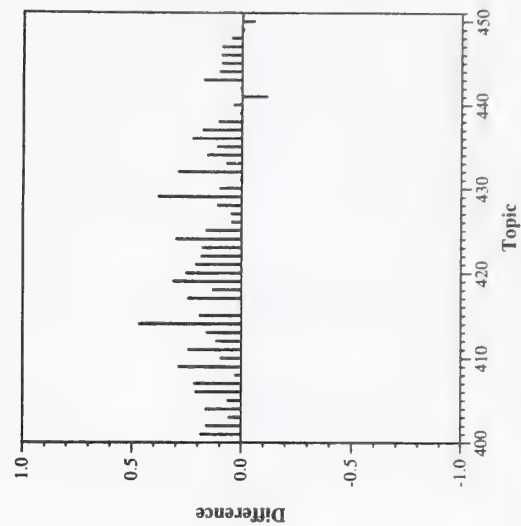
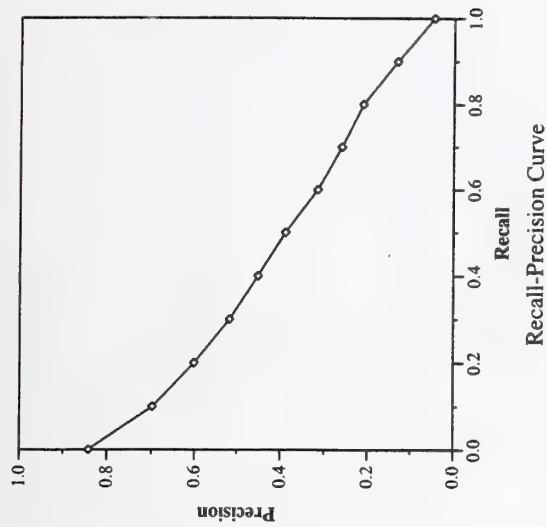


Difference from Median in Average Precision per Topic

Summary Statistics		
Run Number	ok8wmx	
Run Description	content-only runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	2279	
Rel-ret:	2051	

Recall Level Precision Averages	
Recall	Precision
0.00	0.8415
0.10	0.6981
0.20	0.6019
0.30	0.5183
0.40	0.4512
0.50	0.3888
0.60	0.3162
0.70	0.2606
0.80	0.2116
0.90	0.1316
1.00	0.0465
Average precision over all relevant docs	
non-interpolated	0.3829

Document Level Averages	
	Precision
At 5 docs	0.5920
At 10 docs	0.5160
At 15 docs	0.4893
At 20 docs	0.4520
At 30 docs	0.3993
At 100 docs	0.2356
At 200 docs	0.1527
At 500 docs	0.0754
At 1000 docs	0.0410
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4000

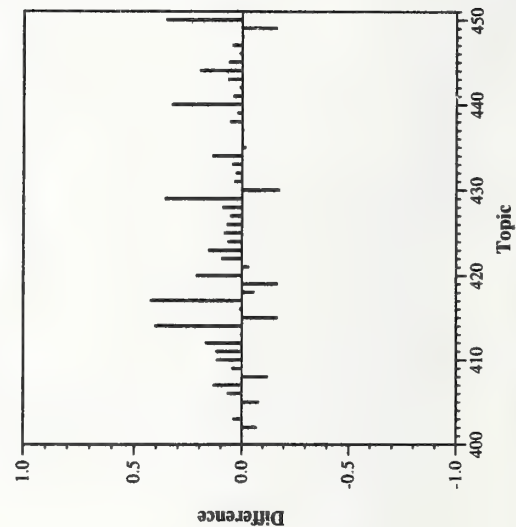
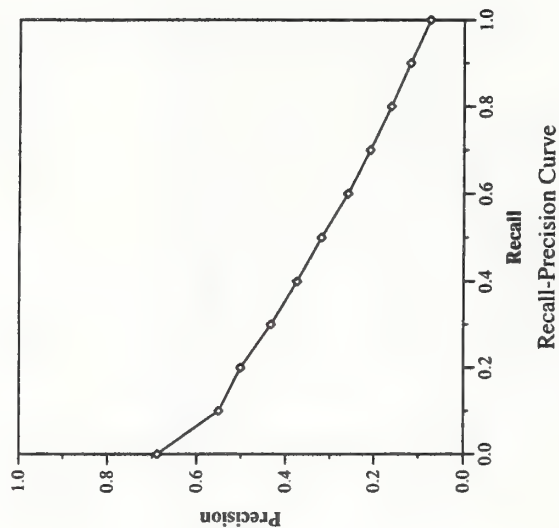


(small) web track results — MultiText Project

Summary Statistics		
Run Number	uwmt8w0	
Run Description	content-only runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	48776	
Relevant:	2279	
Rel-ret:	1795	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6888
0.10	0.5501
0.20	0.5013
0.30	0.4333
0.40	0.3749
0.50	0.3197
0.60	0.2602
0.70	0.2093
0.80	0.1617
0.90	0.1184
1.00	0.0743
Average precision over all relevant docs	
non-interpolated	0.3066

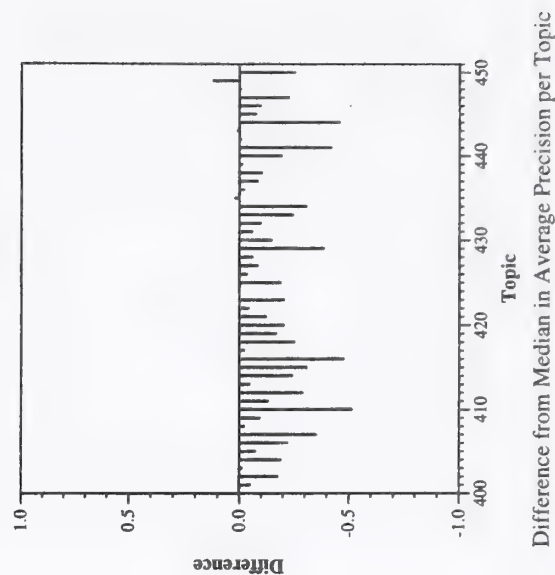
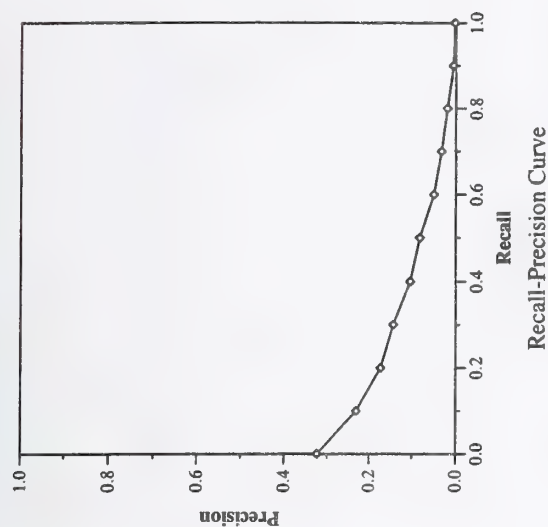
Document Level Averages	
	Precision
At 5 docs	0.4440
At 10 docs	0.4000
At 15 docs	0.3853
At 20 docs	0.3620
At 30 docs	0.3287
At 100 docs	0.2036
At 200 docs	0.1341
At 500 docs	0.0660
At 1000 docs	0.0359
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3170



Summary Statistics		
Run Number	hio1	
Run Description	content-only runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	2279	
Rel-ret:	1292	

Recall Level Precision Averages	
Recall	Precision
0.00	0.3233
0.10	0.2309
0.20	0.1726
0.30	0.1432
0.40	0.1043
0.50	0.0823
0.60	0.0503
0.70	0.0326
0.80	0.0191
0.90	0.0055
1.00	0.0024
Average precision over all relevant docs	
non-interpolated	0.0927

Document Level Averages	
	Precision
At 5 docs	0.1520
At 10 docs	0.1540
At 15 docs	0.1467
At 20 docs	0.1420
At 30 docs	0.1327
At 100 docs	0.0836
At 200 docs	0.0658
At 500 docs	0.0403
At 1000 docs	0.0258
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1223

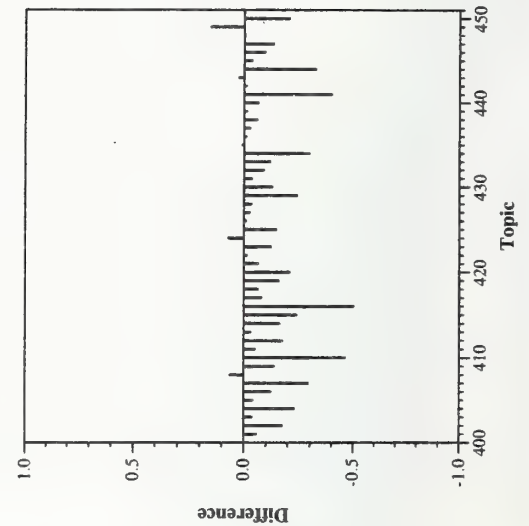
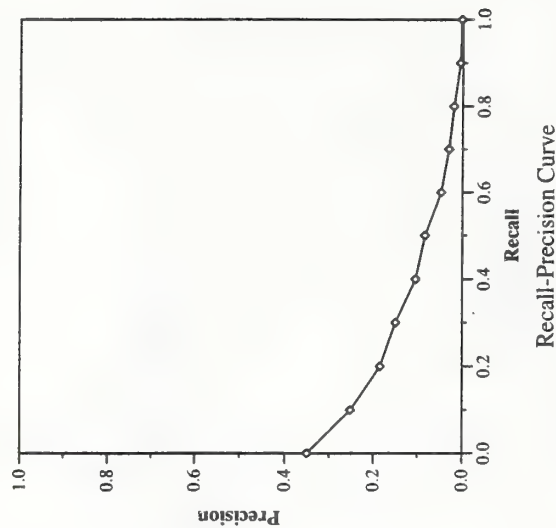


(small) web track results — Oslo College

Summary Statistics		
Run Number	hio2	
Run Description	content-link runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	2279	
Rel-ret:	1288	

Recall Level Precision Averages	
Recall	Precision
0.00	0.3502
0.10	0.2520
0.20	0.1844
0.30	0.1496
0.40	0.1041
0.50	0.0833
0.60	0.0477
0.70	0.0305
0.80	0.0196
0.90	0.0054
1.00	0.0023
Average precision over all relevant docs	
non-interpolated	0.0972

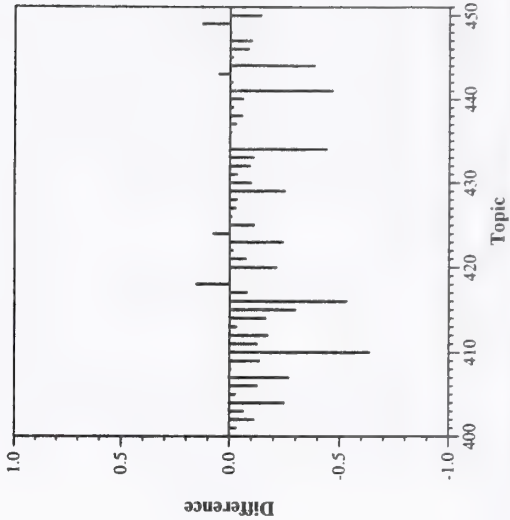
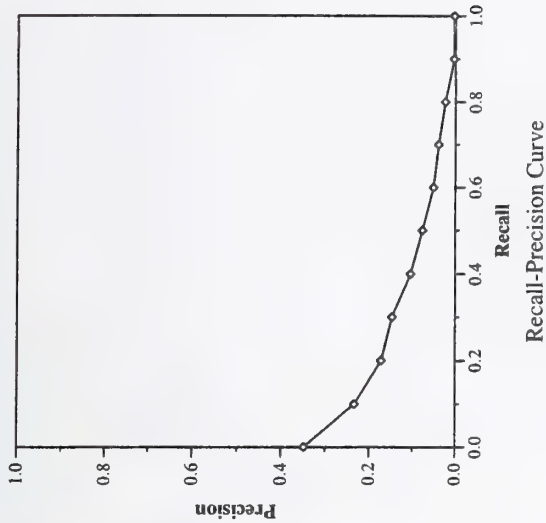
Document Level Averages	
	Precision
At 5 docs	0.1760
At 10 docs	0.1820
At 15 docs	0.1680
At 20 docs	0.1670
At 30 docs	0.1560
At 100 docs	0.0892
At 200 docs	0.0662
At 500 docs	0.0404
At 1000 docs	0.0258
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1314



Summary Statistics		
Run Number	hio3	
Run Description	content-link runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	2279	
Rel-ret:	1394	

Recall Level Precision Averages	
Recall	Precision
0.00	0.3479
0.10	0.2331
0.20	0.1721
0.30	0.1473
0.40	0.1045
0.50	0.0772
0.60	0.0516
0.70	0.0397
0.80	0.0244
0.90	0.0033
1.00	0.0027
Average precision over all relevant docs	
non-interpolated	0.0945

Document Level Averages	
At 5 docs	0.1760
At 10 docs	0.1680
At 15 docs	0.1560
At 20 docs	0.1580
At 30 docs	0.1433
At 100 docs	0.0884
At 200 docs	0.0706
At 500 docs	0.0426
At 1000 docs	0.0279
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1262

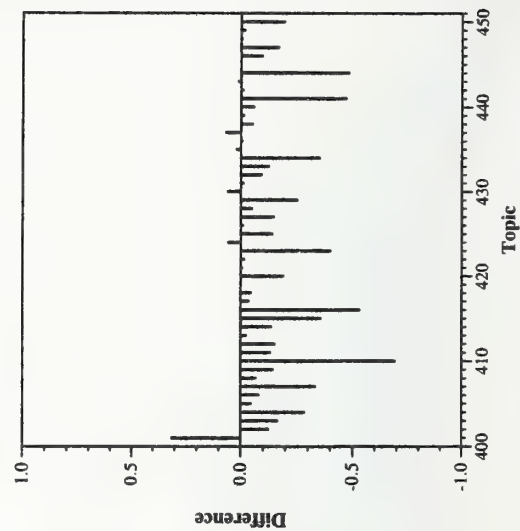
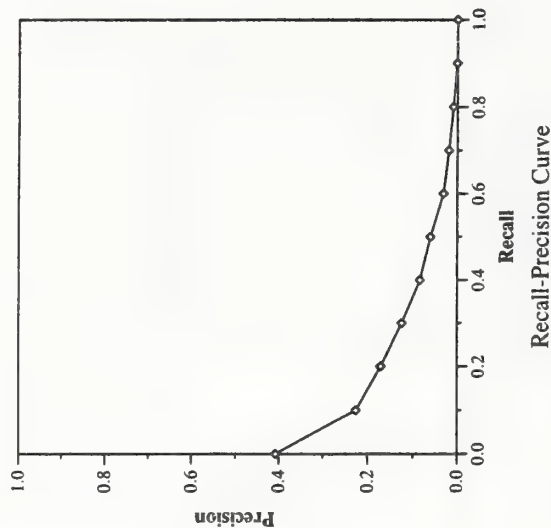


Difference from Median in Average Precision per Topic

Summary Statistics		
Run Number	hio4	
Run Description	content-link runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	2279	
Rel-ret:	1176	

Recall Level Precision Averages	
Recall	Precision
0.00	0.4073
0.10	0.2257
0.20	0.1714
0.30	0.1250
0.40	0.0850
0.50	0.0612
0.60	0.0320
0.70	0.0199
0.80	0.0098
0.90	0.0010
1.00	0.0002
Average precision over all relevant docs	
non-interpolated	0.0859

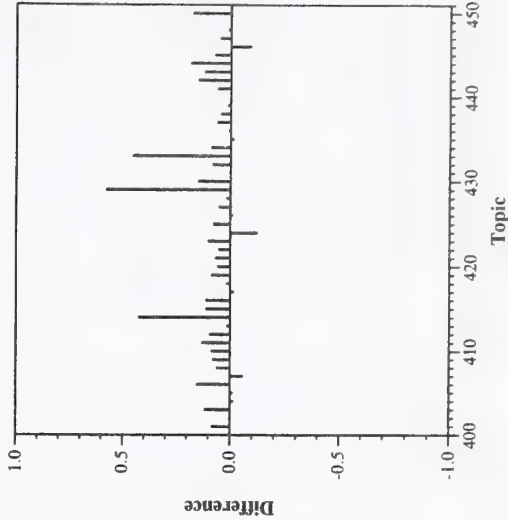
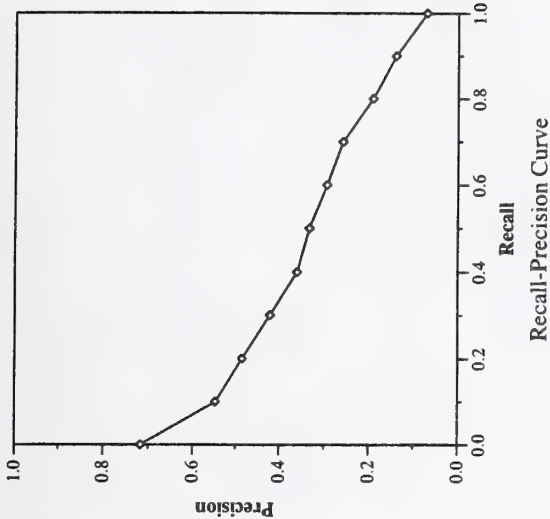
Document Level Averages	
	Precision
At 5 docs	0.2200
At 10 docs	0.2000
At 15 docs	0.1680
At 20 docs	0.1430
At 30 docs	0.1293
At 100 docs	0.0842
At 200 docs	0.0602
At 500 docs	0.0370
At 1000 docs	0.0235
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1230



Summary Statistics		
Run Number	mds08w1	
Run Description	content-only runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	2279	
Rel-ret:	1872	

Recall Level Precision Averages		
Recall	Precision	
0.00	0.7169	
0.10	0.5470	
0.20	0.4851	
0.30	0.4209	
0.40	0.3605	
0.50	0.3334	
0.60	0.2946	
0.70	0.2591	
0.80	0.1915	
0.90	0.1402	
1.00	0.0709	
Average precision over all relevant docs		
non-interpolated	0.3220	

Document Level Averages	
	Precision
At 5 docs	0.4480
At 10 docs	0.4480
At 15 docs	0.3973
At 20 docs	0.3860
At 30 docs	0.3413
At 100 docs	0.1994
At 200 docs	0.1304
At 500 docs	0.0677
At 1000 docs	0.0374
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3585

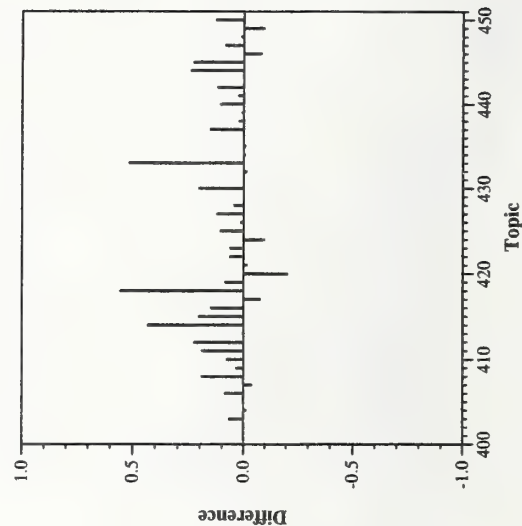
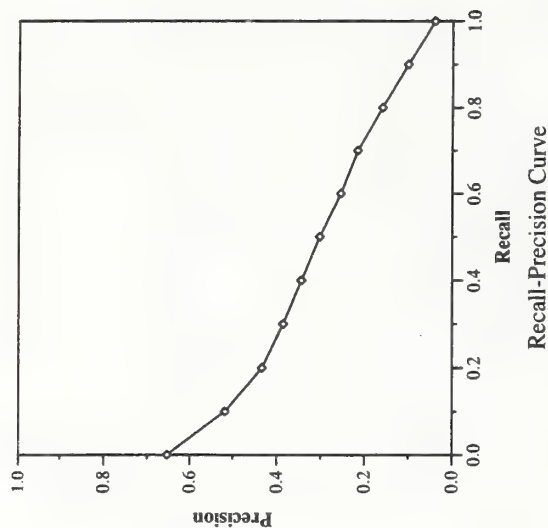


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	mds08w2
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1872

Recall Level Precision Averages	
Recall	Precision
0.00	0.6532
0.10	0.5182
0.20	0.4338
0.30	0.3860
0.40	0.3447
0.50	0.3026
0.60	0.2545
0.70	0.2159
0.80	0.1599
0.90	0.1013
1.00	0.0404
Average precision over all relevant docs	
non-interpolated	0.2878

Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3860
At 15 docs	0.3573
At 20 docs	0.3330
At 30 docs	0.3033
At 100 docs	0.1894
At 200 docs	0.1308
At 500 docs	0.0672
At 1000 docs	0.0374
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3080

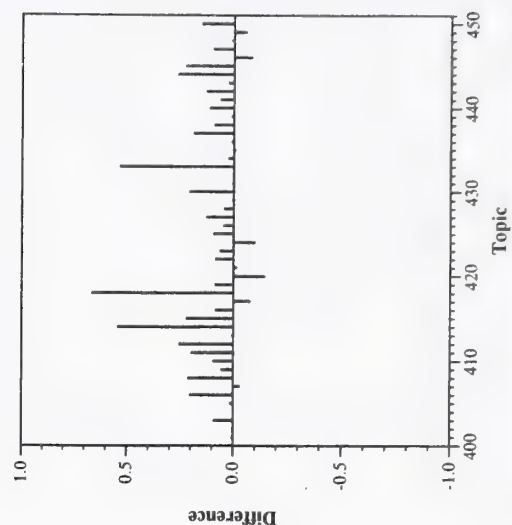
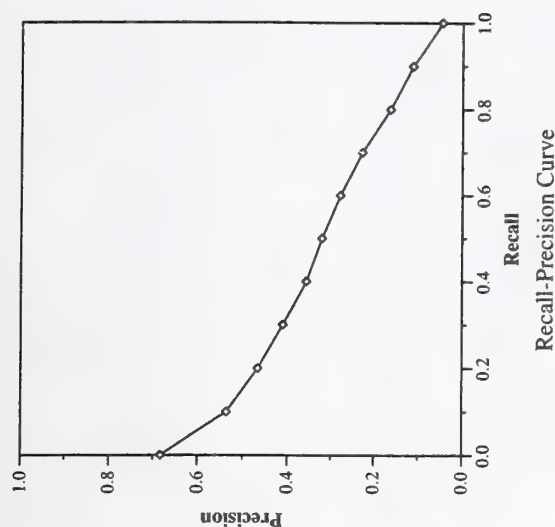


Difference from Median in Average Precision per Topic

Summary Statistics		
Run Number	mds08w3	
Run Description	content-link runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	2279	
Rel-ret:	1878	

Recall Level Precision Averages	
Recall	Precision
0.00	0.6824
0.10	0.5361
0.20	0.4667
0.30	0.4105
0.40	0.3572
0.50	0.3213
0.60	0.2788
0.70	0.2276
0.80	0.1635
0.90	0.1122
1.00	0.0455
Average precision over all relevant docs	
non-interpolated	0.3047

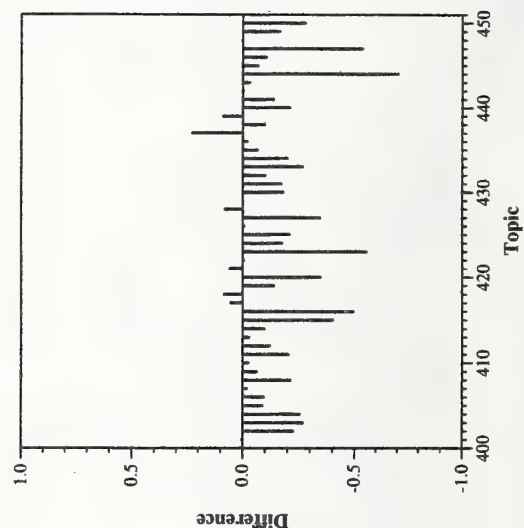
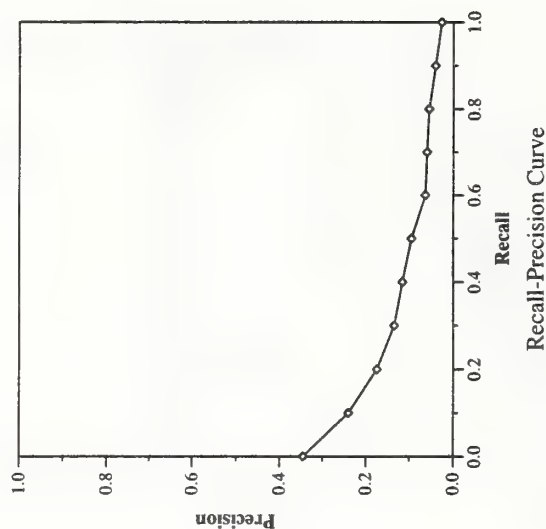
Document Level Averages	
	Precision
At 5 docs	0.4440
At 10 docs	0.4120
At 15 docs	0.3720
At 20 docs	0.3590
At 30 docs	0.3233
At 100 docs	0.2010
At 200 docs	0.1330
At 500 docs	0.0681
At 1000 docs	0.0376
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3320



Summary Statistics		
Run Number	disco2	
Run Description	content-only runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	41444	
Relevant:	2279	
Rel-ret:	1041	

Recall Level Precision Averages	
Recall	Precision
0.00	0.3453
0.10	0.2403
0.20	0.1748
0.30	0.1345
0.40	0.1154
0.50	0.0948
0.60	0.0633
0.70	0.0595
0.80	0.0551
0.90	0.0404
1.00	0.0265
Average precision over all relevant docs	
non-interpolated	0.1023

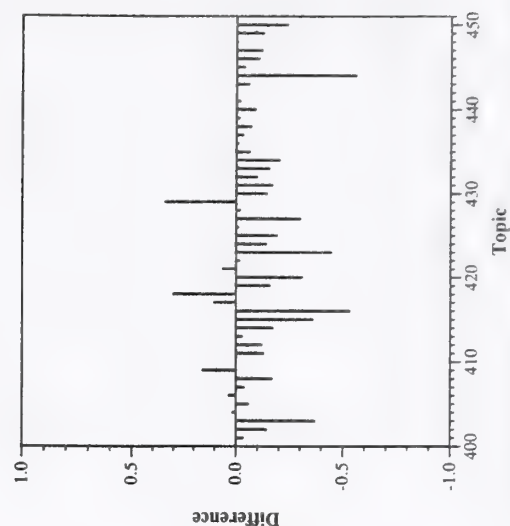
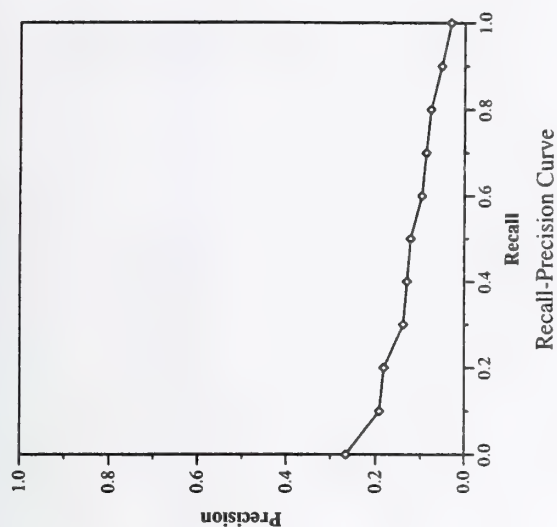
Document Level Averages	
At 5 docs	0.1880
At 10 docs	0.1480
At 15 docs	0.1360
At 20 docs	0.1270
At 30 docs	0.1193
At 100 docs	0.0932
At 200 docs	0.0601
At 500 docs	0.0333
At 1000 docs	0.0208
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1280



Summary Statistics		
Run Number	disco3	
Run Description	content-link runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	40287	
Relevant:	2279	
Rel-ret:	1072	

Recall Level Precision Averages	
Recall	Precision
0.00	0.2651
0.10	0.1916
0.20	0.1817
0.30	0.1391
0.40	0.1306
0.50	0.1229
0.60	0.0976
0.70	0.0882
0.80	0.0778
0.90	0.0532
1.00	0.0315
Average precision over all relevant docs	
non-interpolated	0.1087

Document Level Averages	
	Precision
At 5 docs	0.1240
At 10 docs	0.1120
At 15 docs	0.1133
At 20 docs	0.1110
At 30 docs	0.1020
At 100 docs	0.0770
At 200 docs	0.0598
At 500 docs	0.0353
At 1000 docs	0.0214
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1139

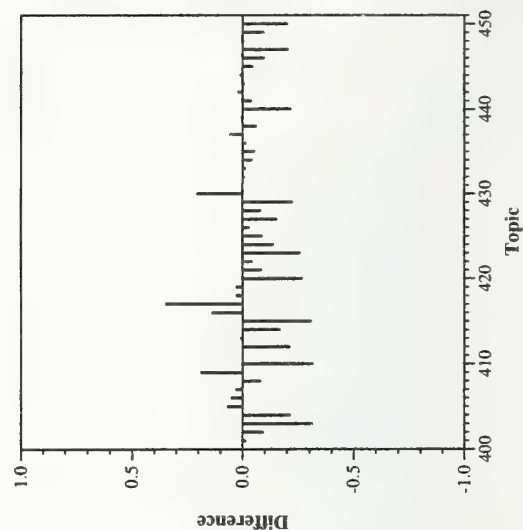
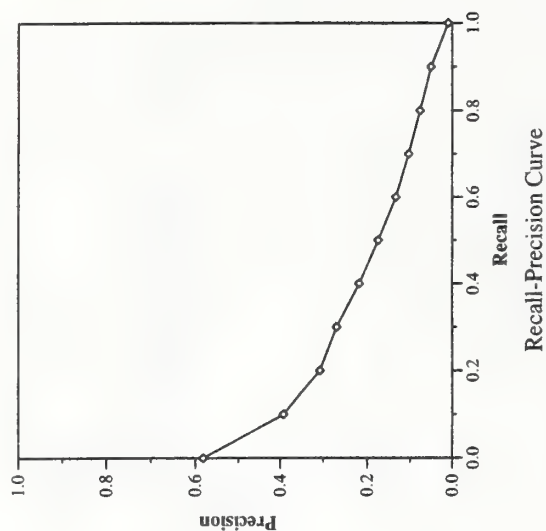


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	Scai8Web1
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1500

Recall Level Precision Averages	
Recall	Precision
0.00	0.5821
0.10	0.3921
0.20	0.3075
0.30	0.2693
0.40	0.2182
0.50	0.1745
0.60	0.1334
0.70	0.1036
0.80	0.0762
0.90	0.0501
1.00	0.0097
Average precision over all relevant docs	
non-interpolated	0.1854

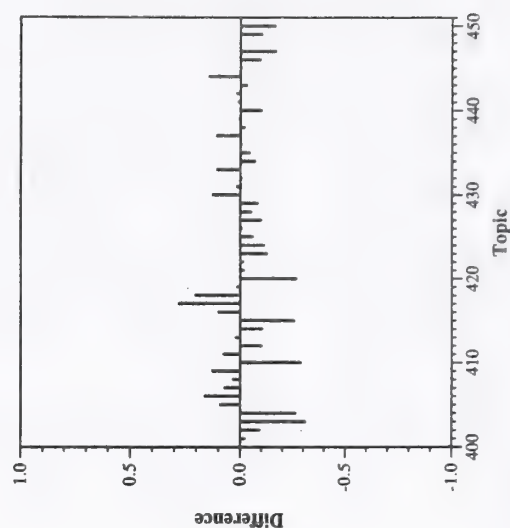
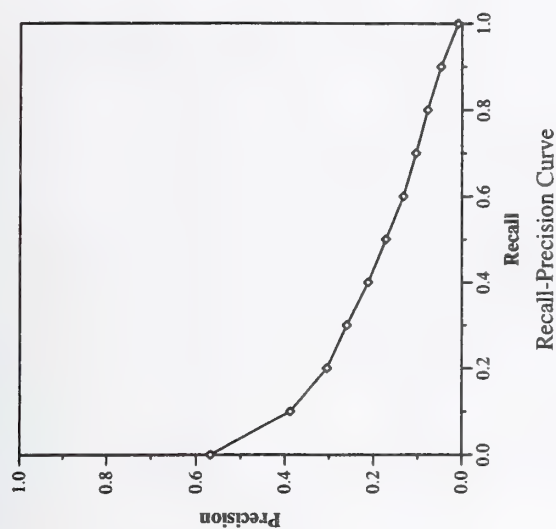
Document Level Averages	
	Precision
At 5 docs	0.3240
At 10 docs	0.3220
At 15 docs	0.2893
At 20 docs	0.2660
At 30 docs	0.2287
At 100 docs	0.1414
At 200 docs	0.0979
At 500 docs	0.0514
At 1000 docs	0.0300
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2235



Summary Statistics	
Run Number	Scai8Web2
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1504

Recall Level Precision Averages	
Recall	Precision
0.00	0.5680
0.10	0.3879
0.20	0.3053
0.30	0.2606
0.40	0.2121
0.50	0.1721
0.60	0.1326
0.70	0.1042
0.80	0.0778
0.90	0.0473
1.00	0.0091
Average precision over all relevant docs	
non-interpolated	0.1819

Document Level Averages	
	Precision
At 5 docs	0.3240
At 10 docs	0.3140
At 15 docs	0.2853
At 20 docs	0.2660
At 30 docs	0.2267
At 100 docs	0.1408
At 200 docs	0.0977
At 500 docs	0.0514
At 1000 docs	0.0301
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2230

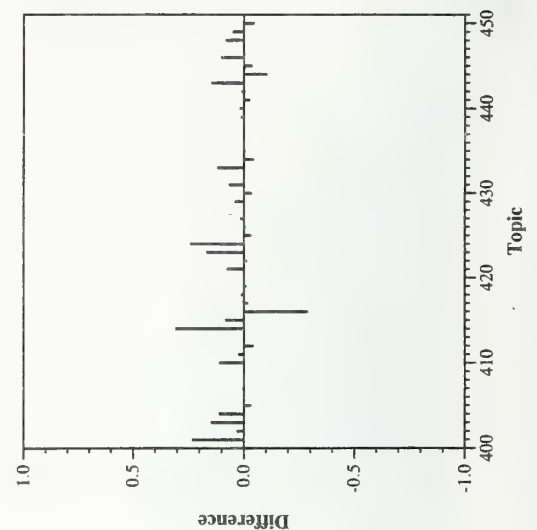
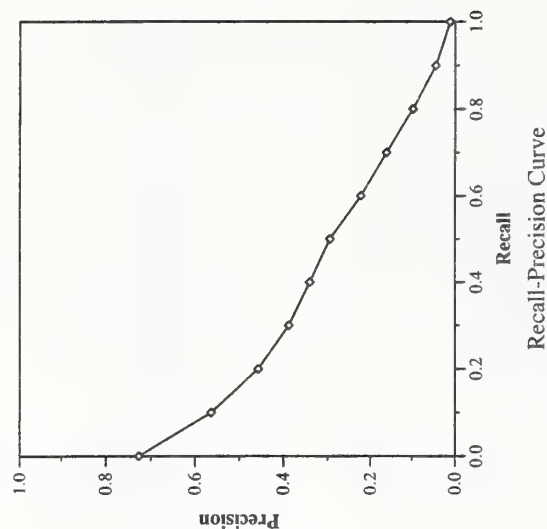


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	UniNEWCt
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1796

Recall Level Precision Averages	
Recall	Precision
0.00	0.7263
0.10	0.5636
0.20	0.4569
0.30	0.3879
0.40	0.3407
0.50	0.2948
0.60	0.2229
0.70	0.1617
0.80	0.0979
0.90	0.0434
1.00	0.0112
Average precision over all relevant docs	
non-interpolated	0.2739

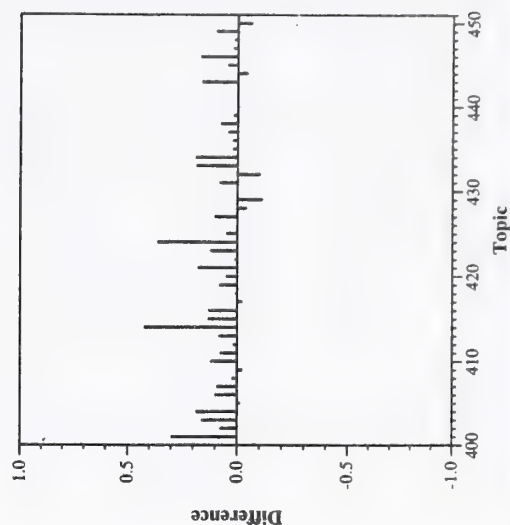
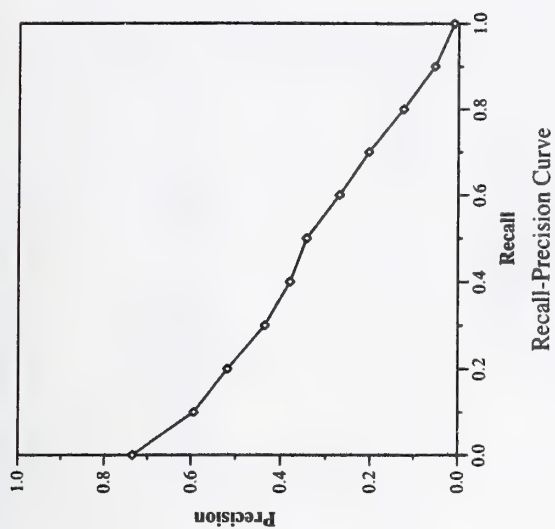
Document Level Averages	
	Precision
At 5 docs	0.4560
At 10 docs	0.4360
At 15 docs	0.4027
At 20 docs	0.3650
At 30 docs	0.3240
At 100 docs	0.1936
At 200 docs	0.1258
At 500 docs	0.0648
At 1000 docs	0.0359
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3256



Summary Statistics	
Run Number	UniNEW2Ct
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1880

Recall Level Precision Averages	
Recall	Precision
0.00	0.7345
0.10	0.5940
0.20	0.5182
0.30	0.4355
0.40	0.3797
0.50	0.3433
0.60	0.2706
0.70	0.2042
0.80	0.1252
0.90	0.0535
1.00	0.0105
Average precision over all relevant docs	
non-interpolated	0.3150

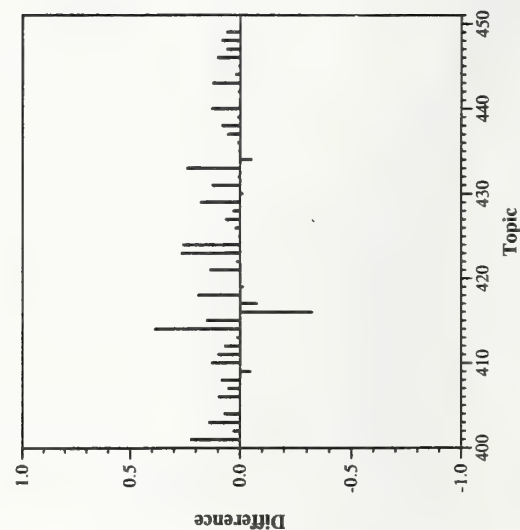
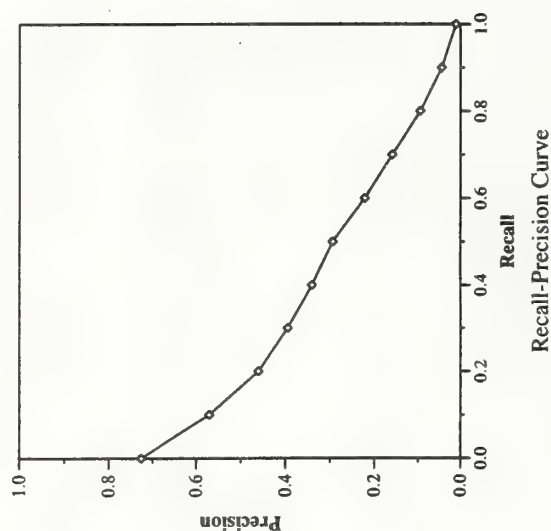
Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4720
At 15 docs	0.4267
At 20 docs	0.3940
At 30 docs	0.3440
At 100 docs	0.2092
At 200 docs	0.1395
At 500 docs	0.0683
At 1000 docs	0.0376
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3383



Summary Statistics	
Run Number	UniNEWLink
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1795

Recall Level Precision Averages	
Recall	Precision
0.00	0.7262
0.10	0.5706
0.20	0.4604
0.30	0.3948
0.40	0.3405
0.50	0.2942
0.60	0.2218
0.70	0.1588
0.80	0.0943
0.90	0.0439
1.00	0.0111
Average precision over all relevant docs	
non-interpolated	0.2747

Document Level Averages	
	Precision
At 5 docs	0.4640
At 10 docs	0.4400
At 15 docs	0.4120
At 20 docs	0.3690
At 30 docs	0.3240
At 100 docs	0.1954
At 200 docs	0.1262
At 500 docs	0.0646
At 1000 docs	0.0359
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3293

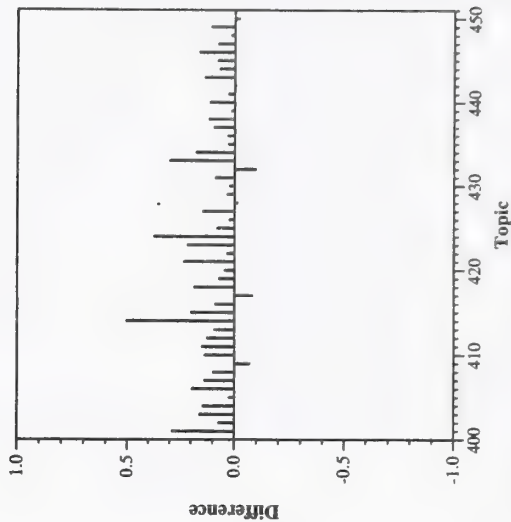
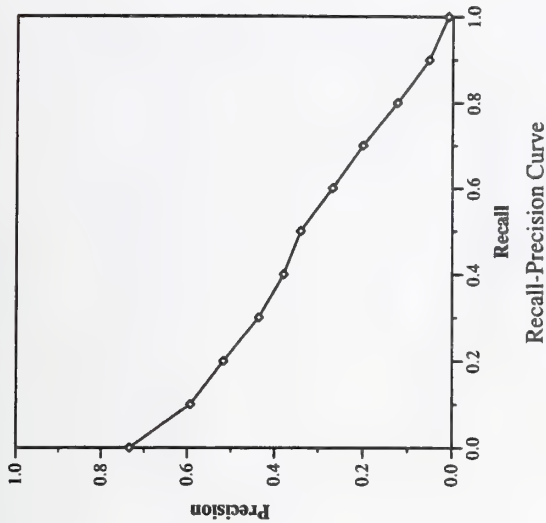


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	UniNEW2Link
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1869

Recall Level Precision Averages	
Recall	Precision
0.00	0.7341
0.10	0.5933
0.20	0.5177
0.30	0.4363
0.40	0.3792
0.50	0.3407
0.60	0.2688
0.70	0.2010
0.80	0.1241
0.90	0.0530
1.00	0.0103
Average precision over all relevant docs	
non-interpolated	0.3137

Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4720
At 15 docs	0.4267
At 20 docs	0.3940
At 30 docs	0.3440
At 100 docs	0.2074
At 200 docs	0.1395
At 500 docs	0.0677
At 1000 docs	0.0374
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3375

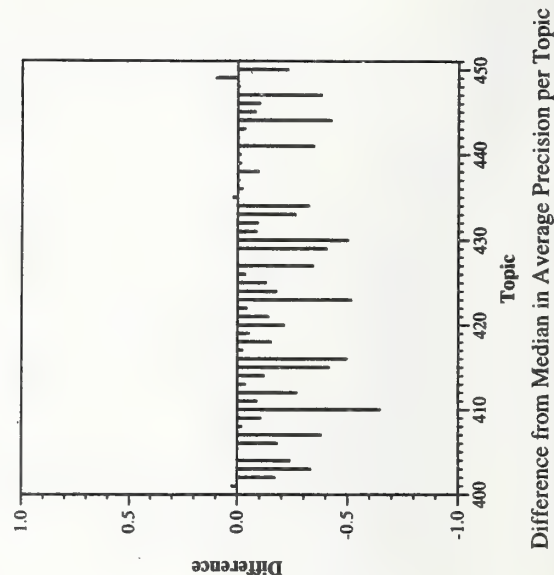
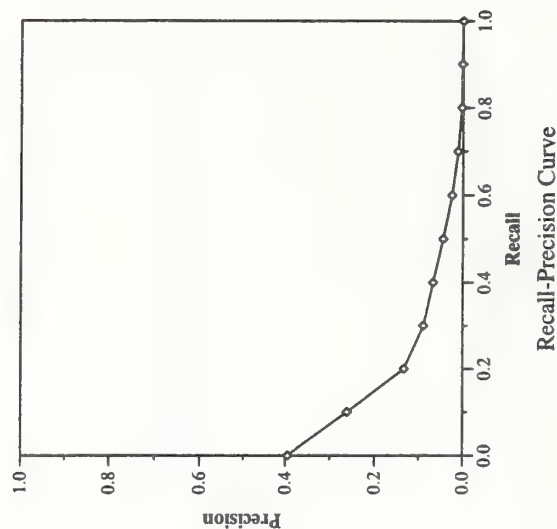


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	uiowaweb1
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1074

Recall Level Precision Averages	
Recall	Precision
0.00	0.3947
0.10	0.2617
0.20	0.1332
0.30	0.0890
0.40	0.0674
0.50	0.0444
0.60	0.0250
0.70	0.0115
0.80	0.0029
0.90	0.0017
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0747

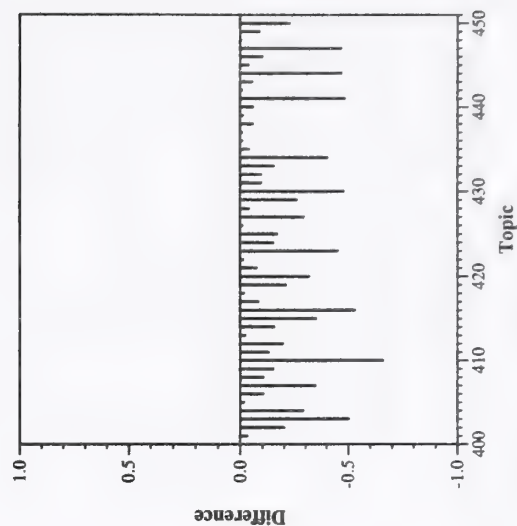
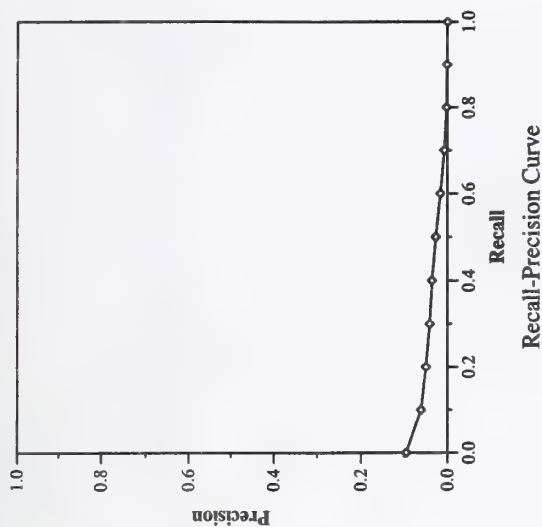
Document Level Averages	
	Precision
At 5 docs	0.2120
At 10 docs	0.1740
At 15 docs	0.1667
At 20 docs	0.1450
At 30 docs	0.1240
At 100 docs	0.0752
At 200 docs	0.0579
At 500 docs	0.0350
At 1000 docs	0.0215
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1077



Summary Statistics	
Run Number	uiowaweb2
Run Description	content-link runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49984
Relevant:	2279
Rel-ret:	1074

Recall Level Precision Averages	
Recall	Precision
0.00	0.0951
0.10	0.0606
0.20	0.0496
0.30	0.0407
0.40	0.0352
0.50	0.0267
0.60	0.0162
0.70	0.0068
0.80	0.0018
0.90	0.0009
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0246

Document Level Averages	
	Precision
At 5 docs	0.0240
At 10 docs	0.0280
At 15 docs	0.0280
At 20 docs	0.0290
At 30 docs	0.0320
At 100 docs	0.0432
At 200 docs	0.0372
At 500 docs	0.0295
At 1000 docs	0.0215
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0383

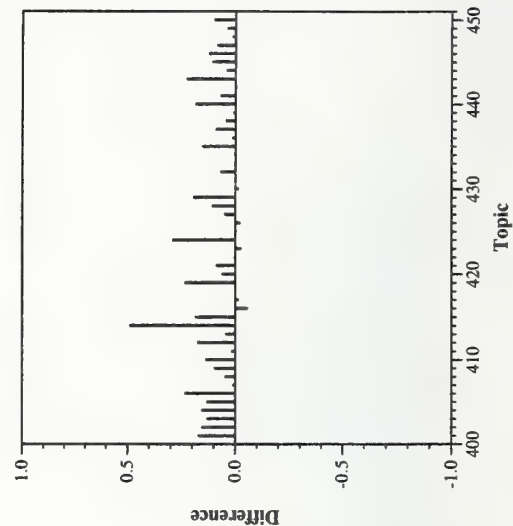
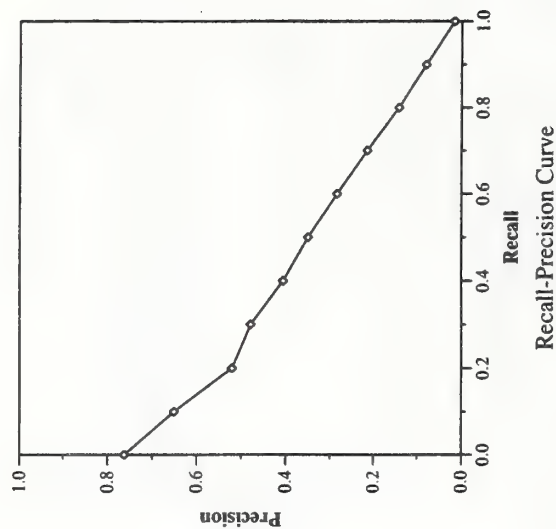


Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	INQ620
Run Description	content-only runs
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	2279
Rel-ret:	1923

Recall Level Precision Averages	
Recall	Precision
0.00	0.7626
0.10	0.6508
0.20	0.5203
0.30	0.4787
0.40	0.4059
0.50	0.3498
0.60	0.2842
0.70	0.2145
0.80	0.1415
0.90	0.0796
1.00	0.0161
Average precision over all relevant docs	
non-interpolated	0.3327

Document Level Averages	
	Precision
At 5 docs	0.5640
At 10 docs	0.5040
At 15 docs	0.4560
At 20 docs	0.4130
At 30 docs	0.3593
At 100 docs	0.2140
At 200 docs	0.1411
At 500 docs	0.0698
At 1000 docs	0.0385
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3471

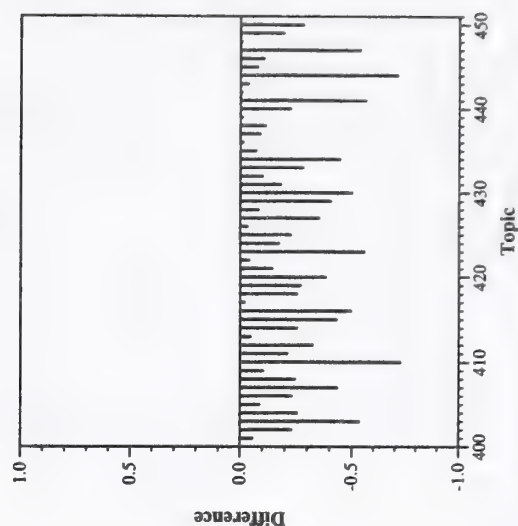
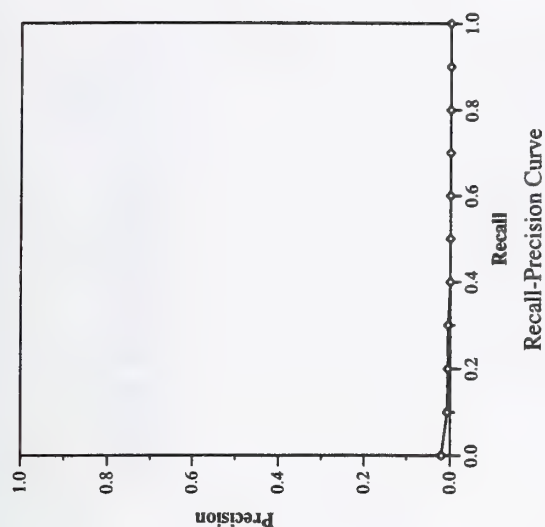


Difference from Median in Average Precision per Topic

Summary Statistics		
Run Number	isw25	
Run Description	content-only runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	50000	
Relevant:	2279	
Rel-ret:	101	

Recall Level Precision Averages	
Recall	Precision
0.00	0.0200
0.10	0.0069
0.20	0.0056
0.30	0.0050
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0018

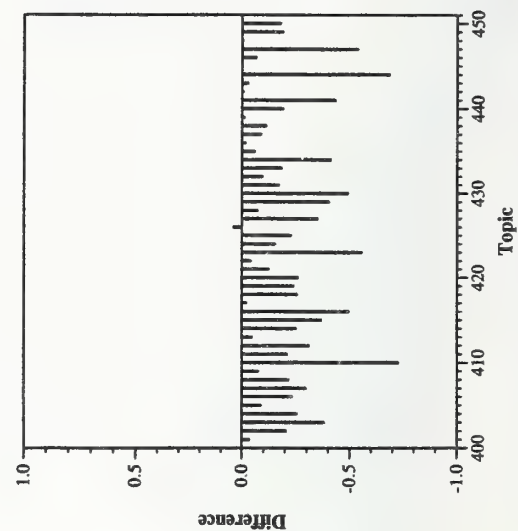
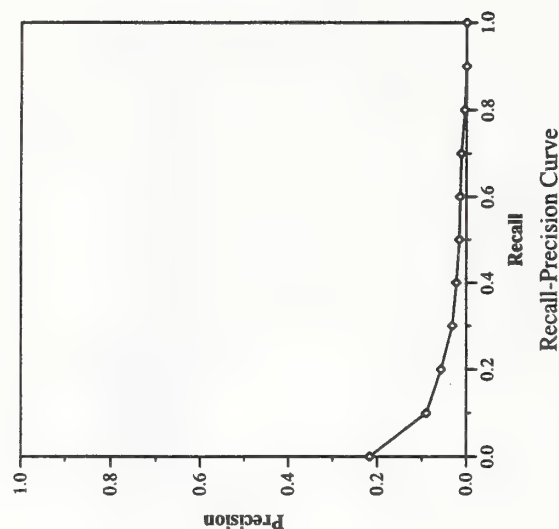
Document Level Averages	
	Precision
At 5 docs	0.0080
At 10 docs	0.0060
At 15 docs	0.0040
At 20 docs	0.0030
At 30 docs	0.0020
At 100 docs	0.0062
At 200 docs	0.0054
At 500 docs	0.0027
At 1000 docs	0.0020
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0048



Summary Statistics		
Run Number	isw25t	
Run Description	content-only runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	49330	
Relevant:	2279	
Rel-ret:	556	

Recall Level Precision Averages	
Recall	Precision
0.00	0.2167
0.10	0.0901
0.20	0.0569
0.30	0.0314
0.40	0.0224
0.50	0.0157
0.60	0.0143
0.70	0.0114
0.80	0.0041
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0291

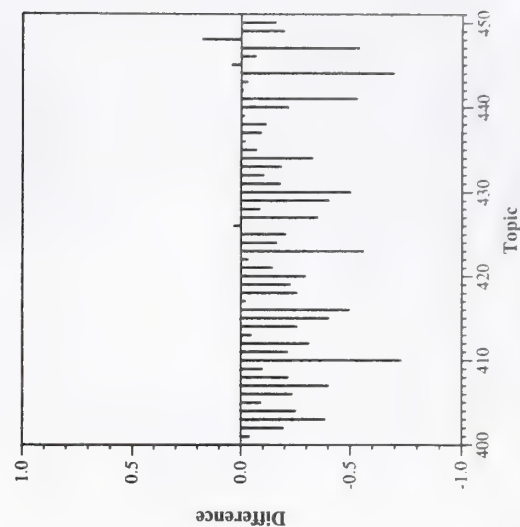
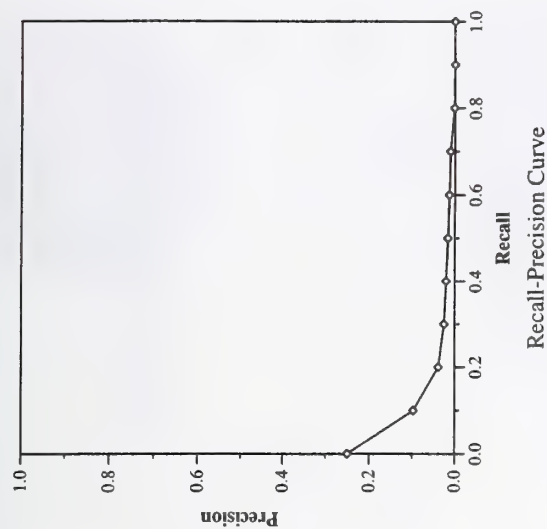
Document Level Averages	
At 5 docs	0.1080
At 10 docs	0.0940
At 15 docs	0.0933
At 20 docs	0.0830
At 30 docs	0.0700
At 100 docs	0.0378
At 200 docs	0.0280
At 500 docs	0.0154
At 1000 docs	0.0111
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0524



Summary Statistics		
Run Number	isw50	
Run Description	content-only runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	49676	
Relevant:	2279	
Rel-ret:	512	

Recall Level Precision Averages	
Recall	Precision
0.00	0.2494
0.10	0.0975
0.20	0.0383
0.30	0.0251
0.40	0.0202
0.50	0.0159
0.60	0.0129
0.70	0.0105
0.80	0.0014
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0291

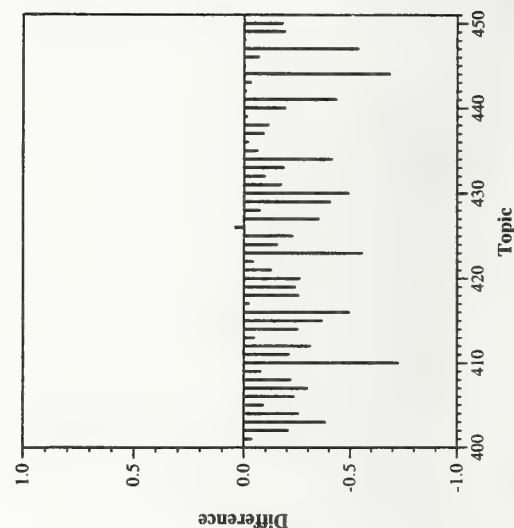
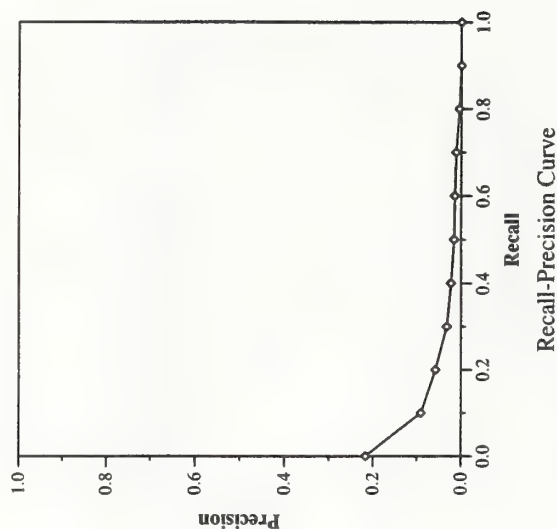
Document Level Averages	
	Precision
At 5 docs	0.0920
At 10 docs	0.0840
At 15 docs	0.0853
At 20 docs	0.0730
At 30 docs	0.0647
At 100 docs	0.0326
At 200 docs	0.0259
At 500 docs	0.0156
At 1000 docs	0.0102
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0451

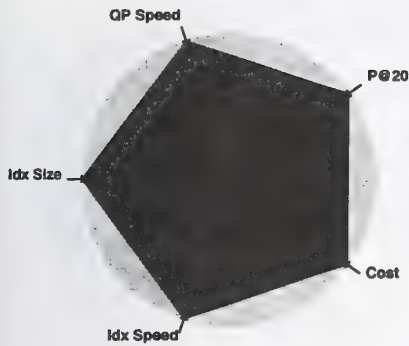


Summary Statistics		
Run Number	isw50t	
Run Description	content-only runs	
Number of Topics	50	
Total number of documents over all topics		
Retrieved:	49330	
Relevant:	2279	
Rel-ret:	556	

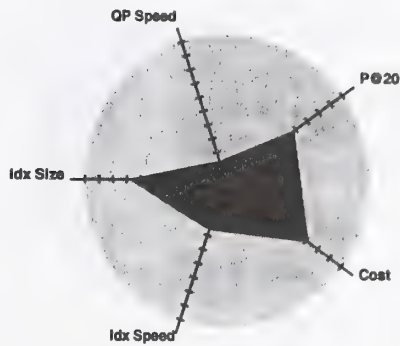
Recall Level Precision Averages	
Recall	Precision
0.00	0.2167
0.10	0.0901
0.20	0.0569
0.30	0.0314
0.40	0.0224
0.50	0.0157
0.60	0.0143
0.70	0.0114
0.80	0.0041
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0291

Document Level Averages	
	Precision
At 5 docs	0.1080
At 10 docs	0.0940
At 15 docs	0.0933
At 20 docs	0.0830
At 30 docs	0.0700
At 100 docs	0.0378
At 200 docs	0.0280
At 500 docs	0.0154
At 1000 docs	0.0111
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0524

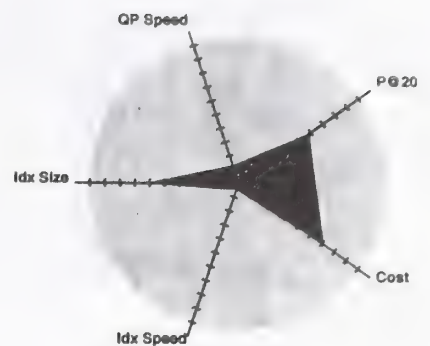




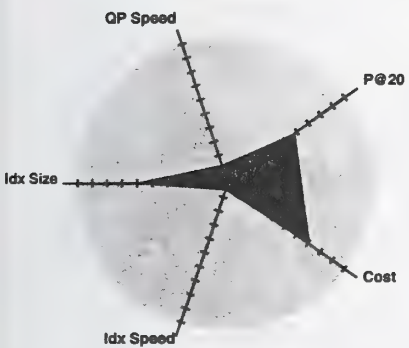
All-Round Best



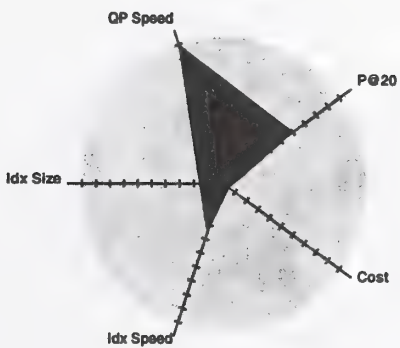
ACSys - acsys8lw0



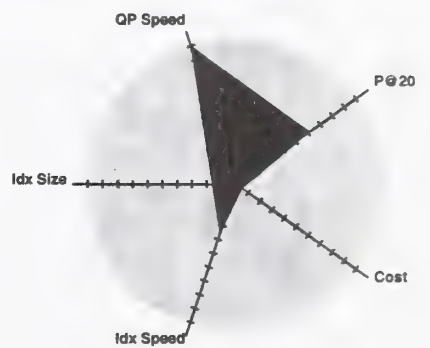
ACSys - acsys8lw0_pr1



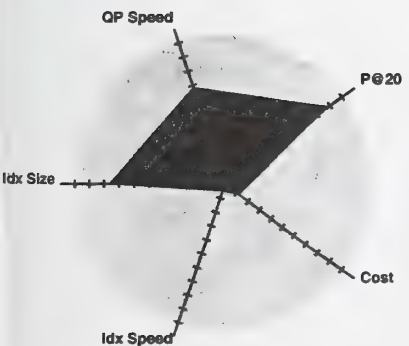
ACSys - acsys8lw0_pr10



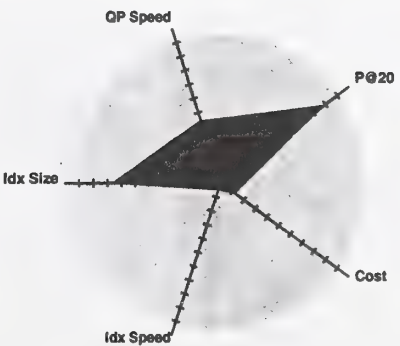
AT&T - att99vici



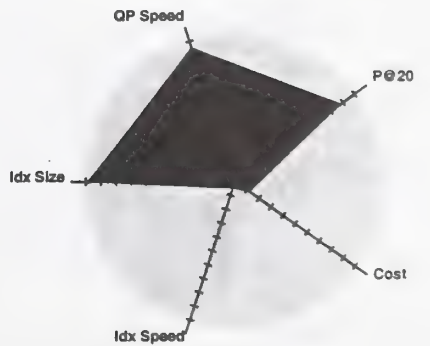
AT&T - att99vlcm



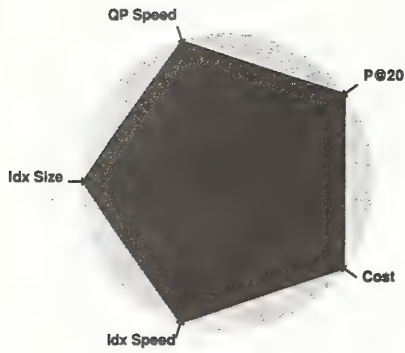
Fujitsu - fl8wlinsb



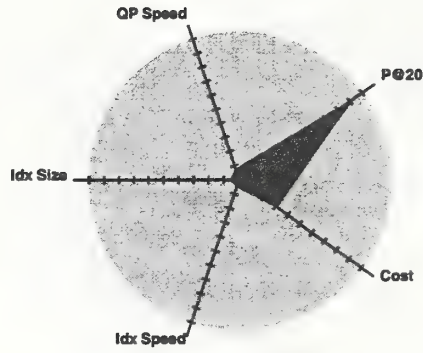
Fujitsu - fl8wlinsr



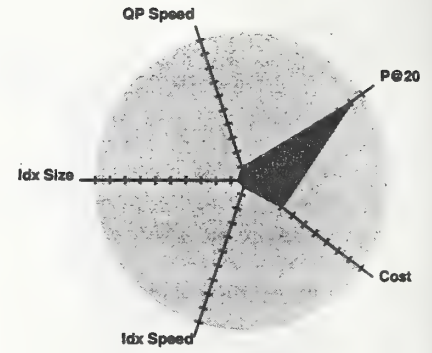
Fujitsu - fl8wlisb



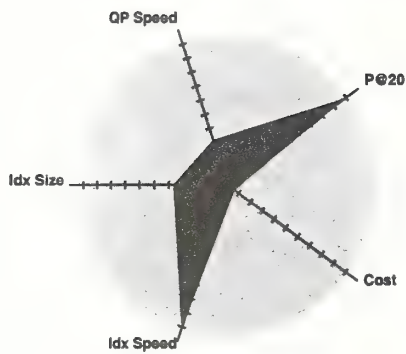
The Perfect System



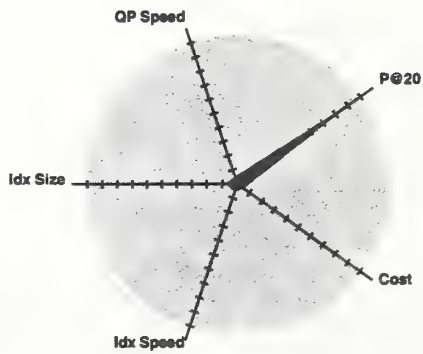
Microsoft - ok8v1



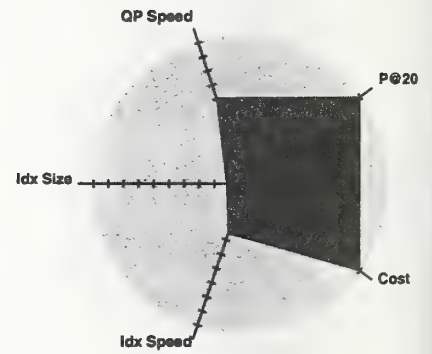
Microsoft - ok8v2



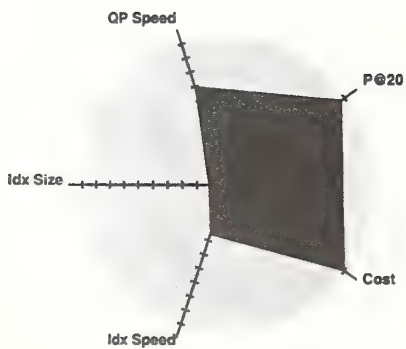
MS/City U - plt8wt1



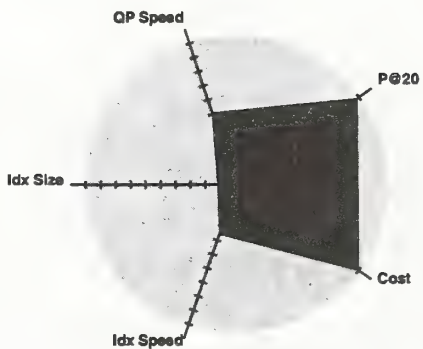
UMass - INQ650



UWaterloo - uwmt8lw0



UWaterloo - uwmt8lw1



UWaterloo - uwmt8lw2

SUMMARY PERFORMANCE COMPARISONS TREC-2 THROUGH TREC-8

Karen Sparck Jones
Computer Laboratory, University of Cambridge

December 8, 1999

The context

This comparison series has attempted to illustrate long-term TREC trends, as embodied in the results for the baseline Adhoc task. As in last year's comparisons, covering TREC-2 - TREC-7, from TREC-5 onwards there has been a more careful separation of different *versions* of the topics, ranging from Very short (titles only) to Long (titles, descriptions and narratives), and between automatic and manual *modes* of query formulation: see the detail given in Table 1.

While last year's comparisons (Appendix B, TREC-7 Proceedings) gave performance details for the whole series from TREC-2 onwards, this year's detail is restricted to TREC-7 and TREC-8 only. First, the way the TREC-6 topics were formed could lead to titles and descriptions that were viewed as complementary rather than as less or more inclusive: this meant that controlled study of the effects of increasing topic length and detail was impossible. In TREC-7 and TREC-8 title terms are included in descriptions (so the difference between descriptions and titles+descriptions is in term frequency for the queries): TREC-7 and TREC-8 therefore supply two cycles of testing on the same topic basis. At the same time, it is evident from the detailed results for these two cycles in Table 2 that there is little difference in performance, whether of best levels or (to a considerable extent) by hardy perennial teams. The TREC-8 results can therefore be seen as a 'wind-up' on the long programme of Adhoc evaluations with the 'traditional' TREC data, and the end of a phase that is also signalled by the fact that evaluation with this type of data is being mothballed for TREC-9.

Table entries

Table 2 follows the same conventions as in previous summaries. Thus the detailed figures are taken from the Working Notes, and cover only the better performing, not all, the teams.

The conventions are as follows: figures are not rounded; performance is assigned to 'blocks'; teams per block are NOT in merit order, but in in Working Notes results order; where there is more than one run per team the best is taken, regardless of the particular strategy used. Simple, hopefully sufficiently identifiable, short names have been given to the teams (with some streamlining where teams have changed name or composition over the years).

TABLE 1 : TOPIC DETAILS

Topic fields available as base for queries, TREC-2 - TREC-7 :

	(TREC-1	TREC-2	TREC-3	TREC-4	TREC-5	TREC-6	TREC-7	TREC-8
T= title	x	x	x		x	x	x	x
D= description	x	x	x	x	x	x	x	x
N= narrative	x	x	x		x	x	x	x
C= concepts	x	x						

Average topic and field length :

Total	107.4	130.8	103.4	16.3	82.7	88.4	57.6	51.8
T	3.8	4.9	6.5	-	3.8	2.7	2.5	2.5
D	17.9	18.7	22.3	16.3	15.7	20.4	14.3	13.8
N	64.5	78.8	74.6	-	63.2	65.3	40.8	35.5
C	21.2	28.5	-	-	-	-	-	-

TABLE 2 : RETRIEVAL PERFORMANCE, TREC-7, TREC-8

TREC ADHOC SEARCH RESULTS FOR PRECISION AT DOCUMENT CUTOFF 30

KEY TO TABLE NOTATIONS :

a = fully automatic searches
m = manual searches

V = very short queries, i.e. title only from topics, aka T
S = short queries description only D
M = medium queries title+description T+D
L = long queries title+description+narrative T+D+N

/contd

	TREC-7 a V	TREC-7 a S	TREC-7 a M	TREC-7 a L	TREC-7 m L	TREC-8 a V	TREC-8 a S	TREC-8 a M	TREC-8 a L	TREC-8 m L
>=60										ManInst
>=55					Clarit					IITetc
>=50					ManInst Waterlo					Oracle
>=45					GMUetc					Clarit GEetc
>=40	NEC	ATT Cityetc UMass	BBN Cityetc NEC UMass	ANU Harris Berkely Toronto	CUNY			ATT FUB Fujitsu IBMTJWs Msoft MIT CUNY	FUB Fujitsu Msoft MIT Neuchat	
>=35	Cityetc CUNY Fujitsu	Cornell Lexis RMIT	ANU Cornell CUNY IRIT Twenty0 Iowa	GEetc Lexis	ATT Fujitsu IBMTJWs Msoft MultTxt RICOH Sab/Crn	UMass		Fujitsu ACSys GEetc IBMTJWg IRIT JHopk MultTxt NTT Sab/Crn UMass Twenty0 Neuchat UMass Twente		
>=30	ATT Cornell CUNY Fujitsu Lexis NEC NTTData RMIT Waterlo	IBMTJWs IRIT	IBMTJWg NTTData Rutgers Berkely UNC	GMUetc FS	ACSys RMIT Twenty0 UMass	IBMTJWs Sab/Crn	ACSys CMU IITetc ImperC JHopk RICOH RMIT Marylnd	RMIT		
>=25	ANU Avignon GEetc IBMTJWg ETH Berkely Marylnd		FUB ImperC JHopk NSA		City/M		UNCy CMU Dartmth			

Performance summary

To give a final overview of performance from TREC-2 - TREC-8, Table 3 gives the highest level of performance reached in each TREC for the various versions and modes.

As this table clearly shows, the early TRECs with 'good' topics reached high levels of performance in both automatic and manual modes; performance in the middle TRECs declined under the much less favourable data conditions (whether of topic information or relevant document accessibility); then in TREC-7 and TREC-8 performance for automatic mode in particular revived. This must be attributed to superior systems, since best manual performance has remained on a plateau. More specifically, amplifying on Tables 2 and 3, it is clear that the better level of performance in the TREC-7 and TREC-8 evaluations was the same.

TABLE 3 : PERFORMANCE SUMMARY

Highest level reached, Precision at Document Cutoff 30, TREC-2 - TREC-8

V	S	M	L	L
T	D	T+D	T+D+N	T+D+N
a	a	a	a	m
>= 65				
>= 60			333	333 888
>= 55				222 777
>= 50			222	666
>= 45				444 555
>= 40 888	777 888	444 777 888	777 888	
>= 35 777				
>= 30 666			555 666	
>= 25	555 666			
>= 20				

Key: 222 = TREC-2 highest performance level, 333 = TREC-3 ditto, etc

(TREC-2 included Concept field

TREC-4 manual did not have Narrative field)

Overall comments

As before, but even more clearly when the evidence of TREC-8 is added in,

1. Many teams obtain similar performance, even at top levels.
2. Manual query formation can give superior performance to automatic, typically reflecting the amount of effort put in and/or user judgements on intermediate outputs.
3. There has been some convergence, especially in automatic searching, on default strategies; but similar performance is also obtained with very different strategies, presumably reflecting the dominating influence of the frequency data that strategies share.
4. Results in TREC generally illustrate the way in which established teams can maintain and enhance their performance; but it also shows that new teams can take advantage of published TREC experience and the rich training data that is available to get up to speed quickly.
5. Performance is broadly correlated with the quality of the topic information available and the difficulty of the topics.
6. However, as the results for TREC-7 and TREC-8 show, it is possible to do almost as well in automatic searching with the minimal (the Very short title) topics as with much longer ones.
7. The best levels of automatic search performance as illustrated by TREC-7 and TREC-8 are quite respectable, and in particular in many cases are achieved with relatively simple, albeit well-motivated, methods. It may be noted that at Cutoff 10, several teams achieved almost 50% Precision in automatic searching even with the Very short titles in TREC-8, and several reached more than 50% with the Medium length titles+descriptions. Manual searching without enormous effort can do better, achieving 70%, but the time and attention required is nevertheless not negligible.

NIST *Technical Publications*

Periodical

Journal of Research of the National Institute of Standards and Technology—Reports NIST research and development in those disciplines of the physical and engineering sciences in which the Institute is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Institute's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Institute's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NIST, NIST annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NIST under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published bimonthly for NIST by the American Institute of Physics (AIP). Subscription orders and renewals are available from AIP, P.O. Box 503284, St. Louis, MO 63150-3284.

Building Science Series—Disseminates technical information developed at the Institute on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NIST under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NIST administers this program in support of the efforts of private-sector standardizing organizations.

Order the following NIST publications—FIPS and NISTIRs—from the National Technical Information Service, Springfield, VA 22161.

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NIST pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NIST Interagency or Internal Reports (NISTIR)—The series includes interim or final reports on work performed by NIST for outside sponsors (both government and nongovernment). In general, initial distribution is handled by the sponsor; public distribution is handled by sales through the National Technical Information Service, Springfield, VA 22161, in hard copy, electronic media, or microfiche form. NISTIR's may also report results of NIST projects of transitory or limited interest, including those that will be published subsequently in more comprehensive form.

U.S. Department of Commerce
National Institute of Standards
and Technology
Gaithersburg, MD 20899-0001

Official Business
Penalty for Private Use \$300